A COST IMPACT ASSESSMENT TOOL

FOR PFS LOGISTICS CONSULTING

THESIS

Angela P. Giddings, First Lieutenant, USAF

AFIT/GOR/ENS/97M-08

19970430 002

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DTIC QUALITY INSPECTED 1

A COST IMPACT ASSESSMENT TOOL

FOR PFS LOGISTICS CONSULTING

THESIS

Angela P. Giddings, First Lieutenant, USAF

AFIT/GOR/ENS/97M-08

THESIS APPROVAL

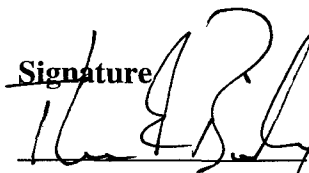**Student:** Angela P. Giddings, First Lieutenant, USAF   **Class:** GOR-97M

Title: A Cost Impact Assessment Tool for PFS Logistics Consulting
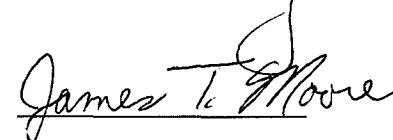
Defense Date: 18 February 1997

Committee:	**Name/Title/Department**	**Signature**
Advisor	Thomas G. Bailey, Lieutenant Colonel, USAF Assistant Professor Department of Operational Sciences	
Reader	James T. Moore, Lieutenant Colonel, USAF Associate Professor Department of Operational Sciences	

AFIT/GOR/ENS/97M-08

A COST IMPACT ASSESSMENT TOOL FOR PFS LOGISTICS CONSULTING

THESIS

Presented to the Faculty of the Graduate School of Engineering

Air Education and Training Command

In Partial Fulfillment of the Requirements for

the Degree of Master of Science in Operations Research

Angela P. Giddings, B. S.

First Lieutenant, USAF

February 1997

The views expressed in this thesis are those of the author and do not reflect the official

policy or position of the Department of Defense or the U. S. Government.

ACKNOWLEDGMENTS

I am grateful for my thesis advisor Lieutenant Colonel Glenn Bailey. Your enthusiasm and sense of humor made researching and writing this thesis a pleasure. Moreover, your even keel when I thought I spotted a storm on the horizon made for smooth sailing all the way. I also appreciate the contributions of my reader, Lieutenant Colonel James Moore. Your insistence on the utmost clarity in every sentence and your close reading of every draft resulted in a written product which is much more polished and readable than it could have otherwise been. Your years of experience in mathematical programming were also a blessing as we attempted to broaden the frontiers of optimality analysis in a mixed integer programming context.

On a more personal note, I am thankful for my AFIT circle of friends. You not only made the rigors of AFIT bearable, but also enriched my life with countless memories which I will always cherish. I thank you for the love and support, the laughter and tears, the joys and fears that we have shared. I am a better person for knowing you all.

Of course, I am also grateful for my husband Scott and my son Aaron. Without your contributions this would have been utterly impossible. Scott, I thank you for cutting short a job you enjoyed and friendships you had just made in Austin in order to come with me to Dayton. I thank you for suffering through a job here which you hated, and still supporting me when I needed you most. Aaron, I thank you for loving me even when I couldn't play football with you, even when I had to work on the computer all day. You both give me safe harbor in rough weather. I love you.

Angela P. Giddings

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Response surface methodology (RSM) is used for optimality analysis of the cost parameters in mixed integer linear programming. This optimality analysis goes beyond traditional sensitivity and parametric analysis in allowing investigation of the optimal objective function value response over pre-specified ranges on multiple problem parameters. Design of experiments and least squares regression are used to indicate which cost parameters have the greatest impact on the optimal objective function value—total cost—and to approximate the optimal total cost surface over the specified ranges on the parameters. The mixed integer linear programming problems of interest are the large-scale problems in supply chain optimization—also known as facility location and allocation problems. Furthermore, this optimality analysis technique applies to optimality analysis of costs or right-hand-side elements in continuous linear programs and optimality analysis of costs in mixed of pure integer linear programs. A system which automates this process for supply chain optimization at PFS Logistics Consulting is also detailed, along with description of its application and impact in their daily operations.

Keywords: Optimality Analysis, Response Surface Methodology, Design of Experiments, Group Screening Experimental Design, Mixed Integer Linear Programming, Supply Chain Optimization, Metamodels.

CHAPTER 1

The research documented in this thesis was sponsored by PFS Logistics Consulting (PFS LC), A Division of PepsiCo, Inc. Analysts at PFS LC assist organizations within the PepsiCo family and external clients in improving their logistics, or supply chain, structure. These projects are called supply chain optimizations and the insights they provide can save millions of dollars for the client organizations. In an effort to streamline the supply chain analysis process and provide added value to their customers, PFS LC decision makers sought a tool for optimality analysis of the cost data used in their supply chain optimization models. Optimality analysis is the broad category of approaches used in mathematical programming to assess how the optimal solution is affected by changes in the model parameters. It includes the traditional sensitivity analysis and parametric analysis techniques as well as approaches which allow you to define a range of investigation on each of many parameters and investigate the optimal objective function value surface over the defined ranges.

Our goal was to provide them with a tool that could indicate the costs with the greatest influence on the optimal solution in a reliable, accessible, expeditious fashion and enhance their optimality analysis capabilities overall. We have done that in a PC-based system which we call the Cost Impact Assessment Tool, or CIAT. CIAT is a loosely connected suite of Turbo Pascal routines, Excel spreadsheets and macros, and JMP statistical software designed to work with the SAILS™ supply chain optimization software.

The remainder of this thesis explains the theoretical underpinnings of CIAT and the method for its implementation. Textual copies of the Turbo Pascal routines and Excel

macros, a description of SAILS™ and JMP as they are used in CIAT, and the base Excel spreadsheet which serves as a user interface are included. The thesis is organized so that Chapter 2 is a stand-alone article on our research, suitable for submission to an academic journal. Chapter 3 outlines some obvious extensions to this work. The appendices contain the details of CIAT and its theoretical background as follows. Appendix A describes the PFS LC environment and details the concerns which led to the development of CIAT. Appendix B contains an extensive literature review which includes explanation of the theory behind the ODS solver engine used in SAILS™ and the techniques applied in CIAT. Appendix C begins with a brief introduction to the SAILS™ package and includes a description of the input and output files in the SAILS™ modeling environment. Appendix D is a detailed outline of the process used when CIAT is applied to a supply chain optimization project. Appendix E reveals the basic Excel spreadsheet which serves as a user interface for CIAT and the Excel macros used to automate some steps in a CIAT application. Finally, Appendix F contains the text of Turbo Pascal routines used in CIAT.

CHAPTER 2

2.1. INTRODUCTION

Supply chain optimization saves millions of dollars each year for a wide array of

industries. Virtually any corporation that obtains raw materials, produces consumer

goods, and delivers them to market is concerned with the efficiency of their logistics

network, or supply chain. Groups such as PFS Logistics Consulting (PFS LC), A Division

of PepsiCo, Inc., specialize in supply chain optimization—developing models of a

business' logistics and recommending changes to improve or optimize efficiency.

Whenever a company contracts a supply chain optimization project with PFS LC,

their work follows the timeline pictured in Figure 2.1 below. (Barlaz, 1996; Clarkson,

1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)
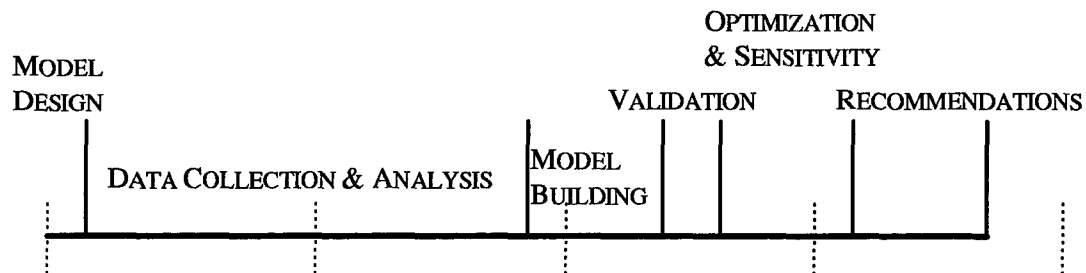
Fig. 2.1. Sample timeline for supply chain optimization project.

We focus on two stages in the timeline—DATA COLLECTION & ANALYSIS, and

OPTIMIZATION & SENSITIVITY. The DATA COLLECTION & ANALYSIS stage clearly requires

the most time in a project; thus, we concentrate on providing a system for streamlining this

stage. The system developed—which we call the Cost Impact Assessment Tool, or

CIAT—also applies to the sensitivity analysis excursions in the OPTIMIZATION &

SENSITIVITY stage. This article gives the theoretical foundation for CIAT and outlines its

2-1

use. It also describes differences between the two stages, and illustrates the use of CIAT through a DATA COLLECTION & ANALYSIS scenario and an OPTIMIZATION & SENSITIVITY scenario performed on a sample supply chain.

2.2. PERTINENT LITERATURE

2.2.1. Supply Chain Optimization.

Supply chain optimization is explored extensively in the mathematical programming literature. (Sengupta and Turnbull, 1996; Arntzen, Brown, Harrison, and Trafton, 1995; Geoffrion and Powers, 1993; Geoffrion and Graves, 1974) Although the term supply chain was coined in the 1990's, these problems have been formulated and solved for decades. Sometimes referred to as the facility location and allocation problem, the supply chain optimization problem is generally modeled as a large-scale mixed integer linear programming problem. (Francis and White, 1974; Love, Morris, and Wesolowsky, 1988) Although economies of scale can be approximated with a piece-wise linear objective function, when they are expected to be significant the model is revised to incorporate a nonlinear objective function. Due to the large number of continuous and binary variables typically involved in a model of this sort, developing efficient solution techniques is another fertile area for research. Methods such as factorization and Benders' Decomposition are proposed and incorporated into solution engines. (Geoffrion and Graves, 1974; Brown, Graves, and Honczarenko, 1987)

2.2.2. Optimality Analysis.

Optimality analysis is another fruitful area for research in mathematical programming. Optimality analysis is the broad category of techniques for investigating how the optimal solution of a mathematical program is affected by changes in the model

2-2

parameters. It includes the traditional sensitivity and parametric analysis techniques. Due to the necessary and sufficient conditions associated with an optimal solution to a (continuous) linear programming problem, the sensitivity of the solution to problem data is easy to compute and is often provided directly by the solver employed. (Bazaraa, Jarvis, and Sherali, 1990; Winston, 1994; Ward and Wendell, 1990) Techniques are also available which show how the optimal solution changes over specified ranges on all of the problem parameters. A particularly applicable approach uses designed experiments to specify an efficient list of problem parameter settings for investigation. Then, response surface methodology (RSM) is used to develop a mathematical model of the surface based on the solutions generated for each scenario in the list. (P. W. Smith, 1975; Johnson, Bauer, Moore, and Grant, 1996)

Unfortunately, the discrete nature of the feasible solution space for a mixed integer linear programming problem complicates the issue of sensitivity analysis. Therefore, there is less literature on sensitivity analysis approaches for mixed integer linear programming. (Geoffrion and Nauss, 1977; Nauss, 1979) We will show that techniques used for analysis of (continuous) linear programming problems can also be applied in mixed integer linear programming. Specifically, the design of experiments/response surface methodology (DOE/RSM) approach can describe the sensitivity of the optimal objective function value (cost) with respect to changes in the cost parameters.

2.3. SPECIFIC PROBLEM

Returning to the project timeline illustrated in Figure 2.1, we describe each stage and explain the need for our system in the DATA COLLECTION & ANALYSIS and OPTIMIZATION & SENSITIVITY stages.

2.3.1. Model Design.

This first stage in the project timeline involves sketching out the network configuration of the business' supply chain, and deciding what level of detail is appropriate for the analysis. The network representation for a simple supply chain is given in Figure 2.2. In the MODEL DESIGN stage, the analyst determines which facilities are in the network and how those facilities are connected; determines the importance of individual production lines; and decides what specific questions are to be answered by the analysis. Some of the questions typically answered through a supply chain optimization project are:

- "How many distribution centers (DCs) are in the best (optimal) solution?",
- "Where should we place our DCs?",
- "Which DCs should each production facility feed?",
- "Which customer zones should be supplied by which DCs, and to what degree?",
- "How many plants are in the optimal solution?",
- "Where should the plants be located?",
- "How should products be allocated to plants?", and
- "Where should raw materials be acquired?".

Although analysts at PFS LC insist that the supply chain must be considered as a whole, rather than by focusing on its individual components or echelons, our CIAT prototype was designed to focus on the DC echelon. (Clarkson, 1997) The DCs are a good starting place since they often provide a great opportunity for savings. Moreover, the client company is often interested in making changes to their DCs even before the PFS LC analyst steps in. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

| Raw Mat. | Production | Distribution | Customer |
| Vendor | Facilities | Centers | Zones |
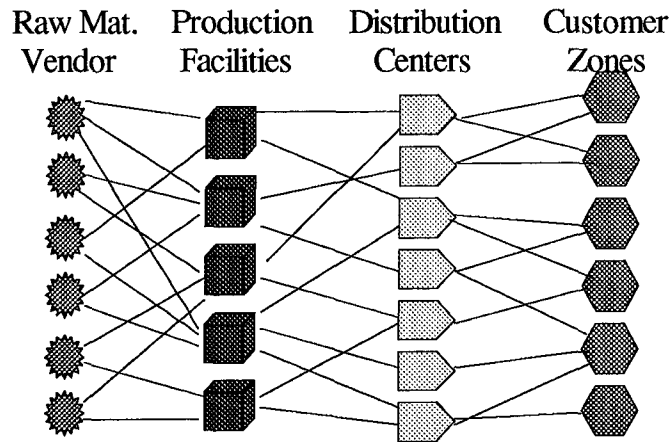
Fig. 2.2. Supply chain network representation.

2.3.2. Data Collection and Analysis.

At this time, data such as costs and distances are developed for the network

model. Although this sounds like a simple endeavor, many of the data elements required

are not readily available. Accounting systems used by the client company may make it

difficult to isolate the variable cost for space at a current distribution center. For instance,

the accountant's fixed cost on a warehouse facility likely includes all space related costs

(i.e. rent). However, for strategic modeling purposes, analysts at PFS LC believe that a

portion of the rent should be considered a variable cost as a function of facility capacity.

(Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

Also, supply chain optimization often evaluates proposed facilities which do not

yet exist. Therefore, the costs for these proposed facilities must be approximated based

upon the expected scale of the facility and its location. In these cases, the costs must be

"engineered" from a zero base, which is a time consuming process. (Clarkson, 1997)

Another approach is to conduct a regression analysis, with facility costs modeled as a

function of facility size. In this case, extensive research may be required to obtain data for

building the regression model. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

2.3.3. Model Building.

In this stage, the data and network representation of the supply chain are combined. The analyst translates the network configuration and data into a mathematical model suitable for solution by a mathematical programming solver. The model generally begins in an unconstrained form, with the analyst incrementally increasing the detail, through addition of model constraints, until the analyst and client are satisfied with the model's level of fidelity. This stage may be revisited later if shortfalls in the model are discovered, or if contacts in the client company are not satisfied with the level of detail included. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

2.3.4. Validation.

This stage ensures that the model appropriately reflects the real supply chain. This generally involves running the model with historical data to verify that the solutions it returns are acceptable in light of the actual system behavior observed, with further refinements made as departures from reality are noted. Again, with the client company involved, the project does not proceed without mutual satisfaction. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

2.3.5. Optimization and Sensitivity.

Finally, useful results are obtained in the supply chain optimization project. The current or projected data are used in the validated model to determine what network configuration is best, and what the expected cost and profit figures are for this

configuration. This stage may also involve consideration of customer service standards for the selected network, and comparisons to current networks or competing configurations of interest to the decision makers. Then, scenarios are run to test the sensitivity of the solution to changes in the input data. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

2.3.6. Recommendations.

In this final stage, results are prepared for presentation to the decision makers. This stage includes determination of alternative solutions which may be preferable to the decision maker for reasons not captured in the optimization model. Also, if many sweeping changes are suggested by the optimal solution, the analyst creates prioritized lists of the changes in order to facilitate a gradual transition and reap the greatest savings for the client. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

The entire process provides an excellent opportunity for the client company to step back and evaluate the way it does business. The greatest benefit occurs if contacts within the client company continue to be energized and motivated throughout the project timeline. Unfortunately, the rigors of gathering the data requested by PFS LC analysts in the second stage, coupled with associated delays in the project, makes it difficult to maintain client enthusiasm. Analysts at PFS LC need a tool to focus and streamline the DATA COLLECTION & ANALYSIS stage of their analysis, in order to prevent enthusiasm from waning. A tool for broadening their optimality analysis capabilities for use in the OPTIMIZATION & SENSITIVITY stage would also increase client involvement. (Barlaz, 1996; Clarkson, 1996; Deviprasad, 1996; Hansen, 1996; Villareal, 1996)

2.4. METHODOLOGY

2.4.1. Theoretical Basis.

Since RSM fits a smooth mathematical function to the results of a designed experiment, it is not automatically thought of as a means to express the optimal solution of a mixed integer linear programming problem, which has discrete jumps in its feasible solution space. However, if the objective function cost parameters are used as the factors in the experimental design and the optimal objective function value (total cost) is viewed as the response, a second-order response surface can provide an appropriate representation.

The quadratic models of RSM have been used for optimality analysis of both the objective function coefficients and the demands and resource levels of the right-hand side vector in a continuous linear programming context. (P. W. Smith, 1975; Johnson, Bauer, Moore, and Grant,1996) Through the following theoretical development, we show that RSM can also be applied to optimality analysis of the cost parameters in a mixed integer linear programming problem.

Lemma 2.1. The optimal value function of the cost coefficients in an integer linear programming problem is continuous, piece-wise linear, and concave. (Geoffrion and Nauss, 1977:459)

Lemma 2.2. (The Weierstrass Approximation Theorem) "Let f be real-valued and continuous on a compact interval $[a, b]$. Then for every $\varepsilon > 0$ there is a polynomial p (which may depend on ε) such that $|f(x)-p(x)| < \varepsilon$ for every x in $[a, b]$." (Apostol, 1974:322)

Theorem 2.1. The optimal value function of a mixed integer linear program as a function of its cost parameters, *f(t)*, can be approximated by a polynomial over pre-specified ranges in the cost parameters.

Proof. The result follows directly from the lemmas. Our pre-specified ranges in the cost parameter form a hypercube. This hypercube is compact since any closed, bounded interval on the real numbers is compact and an intersection of compact sets is compact. By Lemma 2.1, *f(t)* is continuous. Hence, by application of Lemma 2.2, *f(t)* can be approximated by a polynomial over this pre-specified hypercube. ∎

Furthermore, since the true optimal objective function value surface is known to be concave, we would expect a linear or quadratic function to provide an acceptable approximation. However, since the true optimal value function is faceted rather than smooth (i.e. continuously differentiable), the approximation can never be exact. (Johnson, Bauer, Moore, and Grant,1996:49) It is critical to note that the optimal value function of the right-hand-side parameters in a mixed integer linear programming problem does not meet these requirements, so Theorem 2.1. does not justify applying an RSM technique for optimality analysis of these parameters.

In cases where a large number of cost factors are uncertain, statistical techniques such as group screening and specialized experimental designs can identify the most important cost coefficients. (See Box and Draper, 1987; Box, Hunter, and Hunter, 1978; or Montgomery, 1976 for a solid background on experimental design.) Due to our unique knowledge of the response behavior relative to changes in the cost coefficients, group screening is a particularly useful way of targeting the most influential costs. (Watson, 1961; Patel, 1962)

In group screening, related factors are varied simultaneously as a unified group factor in the experimental design. One major concern is that grouping costs together could mask individual factor effects on the response due to cancellation. (Kleijnen, 1987a; Kleijnen, 1987b; Kleijnen, 1975; Mauro, 1986; Mauro, 1983; D.E. Smith and Mauro, 1982) For example, suppose that Factor A causes a tremendous increase in the response. We say that Factor A has a positive effect on the response. If Factor A is grouped with other factors that contribute a proportional decrease in the response (i.e. have a negative effect on the response), that group would be incorrectly deemed unimportant. This cancellation can never occur in our context, since the following theorem shows that no cost coefficient can have a negative effect on the optimal total cost which is our response.

Theorem 2.2. Increasing a cost coefficient value can never have a negative effect on the optimal total cost of a (mixed integer) linear program with minimization objective.

Proof. The proof is by contradiction. Suppose the theorem were not valid, and that cost N (on variable N) has a negative effect on the optimal total cost. There are exactly two possibilities for cost N.

Case i: Variable N is non-basic before cost N is increased. Since cost N has a negative effect on the optimal total cost, increasing cost N from its original value caused a decrease in the optimal total cost. Yet, if variable N is non-basic in the original solution, increasing cost N will not cause variable N to become basic. Hence, cost N will have no effect on the optimal total cost.

Case ii: Variable N is basic before cost N is increased. Once cost N is increased, two things may happen. Either variable N will remain basic or it will be replaced in the basis by variable R. If variable N remains basic, then its increased

cost causes an increase in the optimal total cost. If variable R replaces variable N in the optimal solution once cost N is increased, this is because use of variable R is now less costly than use of variable N. However, if variable R were less costly than variable N before the increase, variable N would not have been in the basis before the cost increase. Hence, the optimal total cost cannot decrease in this case either. ■

A final concern with the use of group screening in this optimality analysis context is detection of quadratic effects. Without some method to gauge curvature, a group of factors which contribute a quadratic effect about the design center would also be incorrectly labeled insignificant. Unfortunately, since our underlying model is the deterministic mathematical program, the standard statistical tests for curvature or lack of linear fit are not appropriate.

Instead, we can use the R^2 value to indicate the level of curvature present when a linear regression model has been fitted to the data. R^2, the ratio of the sum of squares explained by the model to the total corrected sum of squares, reflects the proportion of the variation of the response values explained by the factor levels in the model. Since the only error present in our model is due to lack of fit of the linear regression model, R^2 decreases with increased curvature. No curvature could be measured without the inclusion of some number of center point rows, with all factors set at the midpoint of their specified range. We now turn to the question of how the center point response should be weighted or duplicated in the regression model, in order for R^2 to properly reflect the presence or lack of curvature.

Since the underlying system is deterministic, one objection to including multiple center points is that we are adding information that is already included in the model. However, varying the number of center point rows simply controls the relative weight of the responses presented when fitting the regression model. For instance, if a single center point row is used in all cases, R^2 will not provide a consistent assessment of curvature in the underlying response. Including a single center point row regardless of the experimental design ensures that the R^2 value will be notably lower in a full factorial design than in its fractionated counterparts. This occurs because the single center point response is a smaller proportion of the entire data table in a full factorial design. Therefore, in order to provide a gauge which reflects curvature equally across design choices, the center point response is weighted equally across designs by including copies of the center point row as a function of the design used.

Since we want to assess curvature in each dimension, it makes sense to include a center point row for each factor in a full factorial design. In a fractional factorial design, the number of center point rows should also be fractionated so that the responses observed at corner points and the center point are weighted as they would be in the full factorial design. For example, where three center points are included in a full factorial design on three factors (eight corner points), the half factorial design on three factors (four corner points) suggests one and one-half center points, with one or two actually used.

Another alternative for gauging curvature is to build a regression model without inclusion of any center points and calculate the prediction error at the center point. This error reflects the dollar error present in total cost (optimal objective function value) due to using the first-order regression model. Therefore, a quadratic model should be employed

2-12

if the center point prediction error is unacceptably large. The addition of center points to the model affects only the intercept term in the regression equation; thus, the parameters on the factors indicate the relative importance of each factor or group, regardless of which approach is used to measure curvature.

If either of the two methods for assessing curvature indicate the first-order regression model is an insufficient representation of the optimal objective function value response surface, then a second-order regression model and experimental design are appropriate. The additional optimization runs required by the selected Box-Benken design or face-centered cube central composite design are submitted to the mathematical programming solver and a quadratic model is fit using least squares regression. (Note that we are not concerned with blocking effects since the system is deterministic.)

2.4.2. Practical Considerations.

The DOE/RSM approach to sensitivity analysis of the cost coefficients in a mixed integer linear program is fairly straightforward in its application, as shown in the flow chart at Figure 2.3. First, the cost parameters are selected and upper and lower bounds are specified for the cost parameters of interest.

If thirty-one or fewer costs are selected, they are used as factors in a two-level experimental design. PC software packages are available which list the number of scenarios required for a full factorial experiment on a specified number of factors, or the associated fractional factorial or Plackett-Burman designs. We use the JMP statistical package produced by the SAS Institute, Inc. Once a design is selected from the list of choices, JMP produces a table of -1, +1, and 0 coded factor settings for that design.
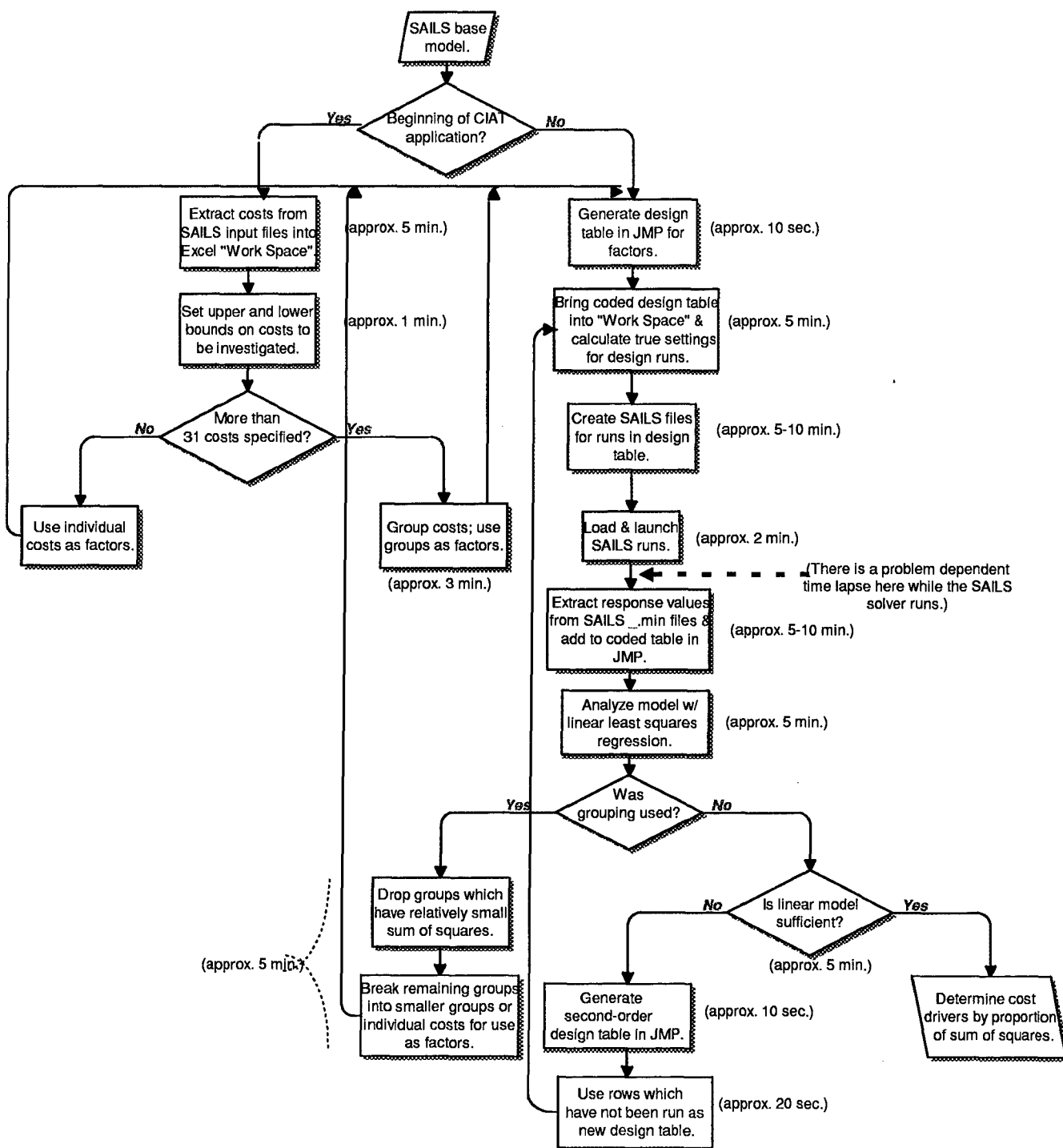
Fig. 2.3. Flow chart for CIAT application.

Each row in the table specifies the cost settings of each factor for a single model

scenario—with -1 denoting the lower bound, +1 the upper bound, and 0 the midpoint.

If more than thirty-one costs are selected for a sensitivity analysis, a grouping strategy is employed. In this case, groups of related costs serve as factors in the experimental design. Thus a -1 setting for a group factor means that all costs in the group should be set at their lower bounds for that scenario; similarly, the +1 and 0 entries for the group factor require setting all costs in the group to their upper bounds and midpoint values, respectively.

Once an experimental design has been generated for the factors selected, the scenarios in that design are submitted to the mixed integer linear programming solver. A center point run—all factors at their 0 level—must be included as a scenario in order to assess the presence of curvature. We apply a solver which is used by PFS LC. Produced by INSIGHT, Inc., and called SAILS™-ODS, it is one of two solvers in a package tailored to supply chain optimization.

In the following passages, we mention the optimal objective function value which is used as the response in our designs. Since the SAILS™-ODS solver actually iterates on the solution until a specified tolerance on optimality is attained, the resulting objective function value is more accurately labeled "near-optimal". However, this is a practical reality with virtually any computer optimization package, and is not a primary issue in our research so we will continue to refer to the solutions SAILS™-ODS provides as "optimal".

The optimal solution is obtained from SAILS™-ODS for each scenario, with its associated optimal cost added to the experimental design table. Before a sum of squares analysis is performed on the design table and optimal costs observed, the center point row should be replicated in the table one time for each factor in the design. Since the data in

the table is deterministic, the center point rows are used to provide a measure of curvature present in the response surface (indicating that additional rows should be added to the design table and the results of the associated optimization runs used to estimate a higher-order functional form for the relationship of the optimal cost to the factors).

With the appropriate number of center point rows in the data table, the statistical package which generated the design is used to perform a sum of squares analysis on the data, through a linear regression or ANOVA procedure with the optimal cost as the response variable and the factor columns as the inputs. The software gives the model corrected sum of squares, the sum of squares for each factor, and the R^2 value for the model.

The proportion of the model sum of squares contributed by each factor is used to determine which factors most influence the response (optimal total cost). If a grouping strategy is in use, factors which contribute a very small proportion of the model sum of squares are dropped from consideration, and the group factors contributing a large proportion are broken into smaller groups of factors which are used in a new design. The R^2 value provided can be used to determine if a quadratic model is justified.

2.5. IMPLEMENTATION

2.5.1. Background.

As mentioned in Section 2.3, the two stages in a supply chain optimization project which benefit from the DOE/RSM approach are DATA COLLECTION & ANALYSIS and OPTIMIZATION & SENSITIVITY. Potential differences between the two areas of implementation are summarized in Table 2.1.

Table 2.1. Differences between two timeline stages of application

Data Collection & Analysis Stage	Optimization & Sensitivity Stage
• Early in the process.	• Late in the process.
• Focused on specific costs.	• Broad-stroke analysis on system.
• Goal: Streamline data collection work.	• Goal: Provide insight for presentation and focus excursions performed in this stage.

Since each supply chain optimization project is slightly different, the applications of CIAT used in practice will more likely fall along a spectrum between the two columns in Table 2.1. For instance, if there is considerable uncertainty about a large number of costs, CIAT application in the DATA COLLECTION & ANALYSIS stage may more resemble the process we will describe as an OPTIMIZATION & SENSITIVITY application in that grouping strategies will be employed. However, designing CIAT to handle the entire spectrum of applications requires a clear delineation of the distinctions shown in Table 2.1.

Ranges on Costs. In DATA COLLECTION & ANALYSIS, the concern is with a small collection of costs that are subject to error or difficult to obtain. Therefore, the ranges on these costs are generally set to reflect the level of uncertainty associated with each individual cost value. In contrast, the analyst often wants to vary many of the model costs in the OPTIMIZATION & SENSITIVITY stage by some fixed percentage of their value to provide a comfort level about the results.

Need for Grouping. Another associated difference is the use of grouping. Analysis in the OPTIMIZATION & SENSITIVITY stage more likely involves large numbers of cost parameters. However, there are cases with widespread uncertainty in the cost data resulting in the need for grouping in the DATA COLLECTION & ANALYSIS stage.

Number and Nature of Responses Investigated. The nature of the responses investigated and functional form used will also differ between the two stages. In the DATA COLLECTION & ANALYSIS stage, the goal is to gauge the importance of individual costs to the bottom line. As a result, the optimal total cost is generally the only response of interest. In contrast, in the OPTIMIZATION & SENSITIVITY stage, additional responses such as the number of distribution centers in the optimal solution or the total distance in the distribution network may be considered.

Fit Required of Regression Model. In DATA COLLECTION & ANALYSIS investigations, the analyst will probably stick to a linear design and model unless the R^2 value is quite low (say below 0.8). In OPTIMIZATION & SENSITIVITY investigations, a high fidelity functional relationship between a selected group of cost parameters and the optimal total cost may be desired for presentation purposes.

2.5.2. Sample Supply Chain.

PFS LC provided a sample supply chain problem along with the associated input data files for the SAILS™-ODS solver. The single-product supply chain is composed of four production facilities (plants), seventeen current and twenty-seven proposed distribution centers (DCs), and 378 customers. The underlying geography is western Pennsylvania and portions of West Virginia and New York.

This supply chain optimization problem focuses on selecting the most efficient collection of distribution centers and routes through them. Distribution center fixed costs are those costs incurred simply by opening the facility, regardless of throughput. Variable costs for each distribution center are variable with respect to throughput volume. Thus,

any cost which can be calculated per cubic foot of warehouse space or in time spent handling product is included in variable costs.

The two sections that follow explain how CIAT could be applied to this sample supply chain in the two stages of the project timeline—DATA COLLECTION & ANALYSIS and OPTIMIZATION & SENSITIVITY.

2.5.3. Scenario 1: Application in Data Collection & Analysis.

All costs associated with the proposed DCs are ranged to investigate their impact on the optimal objective function value. The proposed DCs are selected because their costs are particularly subject to error and time-consuming to refine. For the twenty-seven proposed DCs this amounts to twenty-seven fixed costs and twenty-seven variable costs. A range of plus and minus twenty percent is specified on the variable costs and a range of plus and minus ten percent is used on the fixed costs. Since JMP does not generate a screening design for fifty-four factors, the fixed costs are grouped as factor 1 and the variable costs as factor 2 in the first stage of a screening process.

The two group factors are investigated through a full factorial design with four corner points and two center points included. The regression model fit is first-order with an interaction term. The R^2 value is 0.958 and the adjusted R^2 is 0.895, so the curvature is determined to be negligible. The model sum of squares is 124184.75 with factor 1 (the fixed cost group) contributing 104006.25, or 83.75%. Thus, the variable costs are dropped from the investigation and the twenty-seven fixed costs are retained.

Since only twenty-seven costs remained, we could proceed to individual cost factors in a twenty-eight run Plackett-Burman experiment with additional center point(s). Another option would be to regroup the fixed costs and eliminate more costs before

testing them individually. Proceeding to the Plackett-Burman would provide a more sensitive assessment of all the fixed costs and avoid the necessity of another iteration through the process, but using another grouping stage could save several SAILS™ runs overall.

When we proceed to the Plackett-Burman, it is so highly fractionated, that only one center point run is included. As a result, the R^2 value is 1.000 and the adjusted R^2 is undefined. Using our bottom-line method of assessing curvature, the center point run is not included in the regression. Then the prediction error at that point is evaluated in terms of a tolerance on error in the bottom-line (optimal total cost). In this case, the prediction error amounts to $29,960 in a bottom line of approximately $58,000,000, so it is considered negligible. The regression model fit is a simple linear model, with no interaction terms. The model sum of squares is 220974.96, with five of the cost factors contributing at a notably higher level than any other costs. These five proposed DCs are depicted in Figure 2.4 by letters, while the other proposed DCs are shown as blocks. The greatest contributor is at Mercer (R) with a sum of squares of 95238.89 (42.19%) followed by State College (S) with 51000.89 (23.08%), Bradford (B) and Warren (W) with 25260.04 (11.43%) each, and Meadville (A) with 12900.04 (5.84%). No other fixed cost contribute more than 505.75 to the model sum of squares. Hence, it is our conclusion that since the optimal total cost solution is most sensitive to changes in these five fixed costs over the ranges specified, data refinement resources should be focused on these five facilities.

To further investigate the relationships between these five fixed costs and the optimal system cost, we use a second-order central composite design on these five factors,

with the axial points set on the face of the hyper-cube (that is, the axial settings are at the

coded -1 and +1 levels, requiring only the upper and lower bound values specified). This

design requires thirty-three experimental runs. However, since seven of these are center

point rows, we really only need to perform the remaining twenty-six optimization runs,

then add the center point rows from our previous results. Most DATA COLLECTION &

ANALYSIS CIAT applications would stop before fitting a second-order regression model.

In some cases, though, this regression model can provide useful insight into the system.
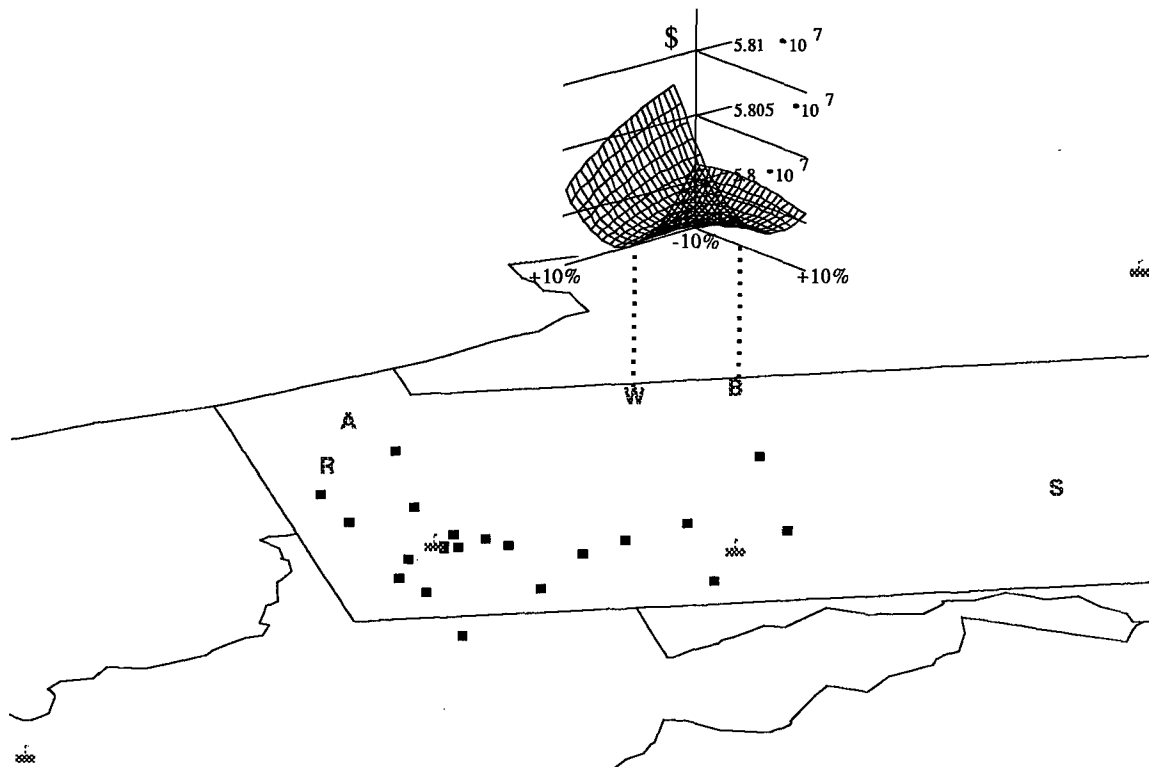
Fig. 2.4. Results of DATA COLLECTION & ANALYSIS scenario.

We find that fitting the second-order model certainly provides interesting results

in this case. First, the second-order regression model reveals a substantial interaction

between the Warren and Bradford fixed costs. This interaction contributes 14460.06 to

the model sum of squares 242544.19. Furthermore, the quadratic term in the regression model reflecting the cost at Warren clearly defines a *convex* surface (Figure 2.4). However, in Section 2.4.1 we state that the optimal value function of a mixed integer linear program is *concave*. This contradiction is explained by considering the optimality tolerance used in SAILS™-ODS.

Recall that our response is actually "near-optimal" due to the iterative method used by the solver to tackle these computationally complex problems. In a central composite design without replications, there are exactly three points used to estimate the form of each of the quadratic terms in the regression model. The incorrect quadratic term in Figure 2.4 occurs because the response values with Warren's fixed cost at both the -10% and midpoint values (and all other costs are held at their midpoint) is $57,978,000, while the response value with Warren's fixed cost at the +10% level is $58,091,000. The value at -10% is the likely culprit, with the solver stopping at this value when a slightly lower response value exists as the true optimal.

There are two possible courses of action when something like this occurs. The first alternative is to re-run the entire design, or at least the disputed points, with a tighter optimality tolerance set. We chose a second alternative, which involves running axial points outside the hyper-cube boundary. The underlying idea is that by extending our sample range we will more likely capture the concave nature of the optimal objective function value. As shown in Figure 2.5, pushing these axial points further from the center illuminated the true concave nature of the response.
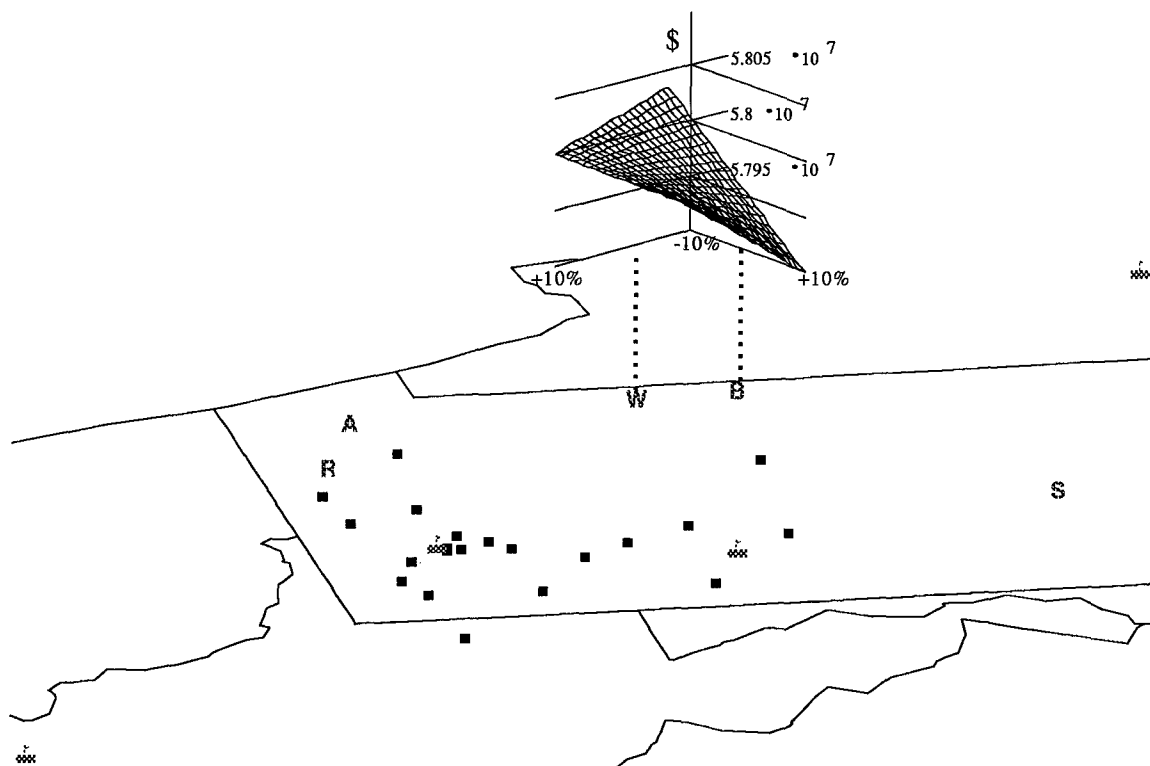
Fig. 2.5. Revised results of DATA COLLECTION & ANALYSIS scenario.

2.5.4. Scenario 2: Application in Optimization & Sensitivity.

The potential variations in applying CIAT in this stage are virtually unlimited.

Depending upon the particular problem, the analyst may want to investigate the

relationships between DC costs, transportation costs, and the response optimal system

cost. Also, the analyst may want to consider alternative responses such as total network

transportation mileage or some customer service measures. Grouping can be used to

investigate the sensitivity of the solution to cost variation along geographic lines or by

managerial units.

We perform an OPTIMIZATION & SENSITIVITY excursion on the sample supply

chain optimization problem provided. Our goal is to investigate how fixed and variable

distribution center costs affect the optimal network solution. Since eighty-eight costs are

investigated, a geographical grouping strategy is employed. A grid is specified in latitude and longitude as shown in Figure 2.6. The fixed and variable DC costs are grouped according to the grid. The letters in Figure 2.6 label each group, and the numbers reflect the number of DCs in each group.

In this scenario, we are not grouping costs in order to screen out those with little impact. Rather, we are placing costs into groups because we would expect their future increases and decreases to be correlated (i.e. cost increases in Pittsburgh will likely affect all facilities in the Pittsburgh area). The goal is to investigate the robustness of our model to variations in costs by looking at the persistence of our recommended solution, simultaneously gaining insight into alternative solutions which would not otherwise have been explored.

For the first stage in this scenario, ranges of plus and minus 10% are set on the fixed and variable costs of all forty-four DCs. Since these costs are grouped into eleven zones according to Figure 2.6, a Plackett-Burman design with twelve corner point runs is appropriate. In order to more thoroughly sample the design space, we adopt the foldover technique from experimental design literature (see Myers and Montgomery, 1995:172-173), by multiplying each coded factor in the Plackett-Burman design by -1, giving a twenty-four corner point design. A center point optimization run is also performed, since we are interested in knowing the active DCs in each solution, rather than the optimal system cost collected in the DATA COLLECTION & ANALYSIS scenario. In the second set of twenty-five runs, the same coded design table is used with costs ranging 30% in both directions.
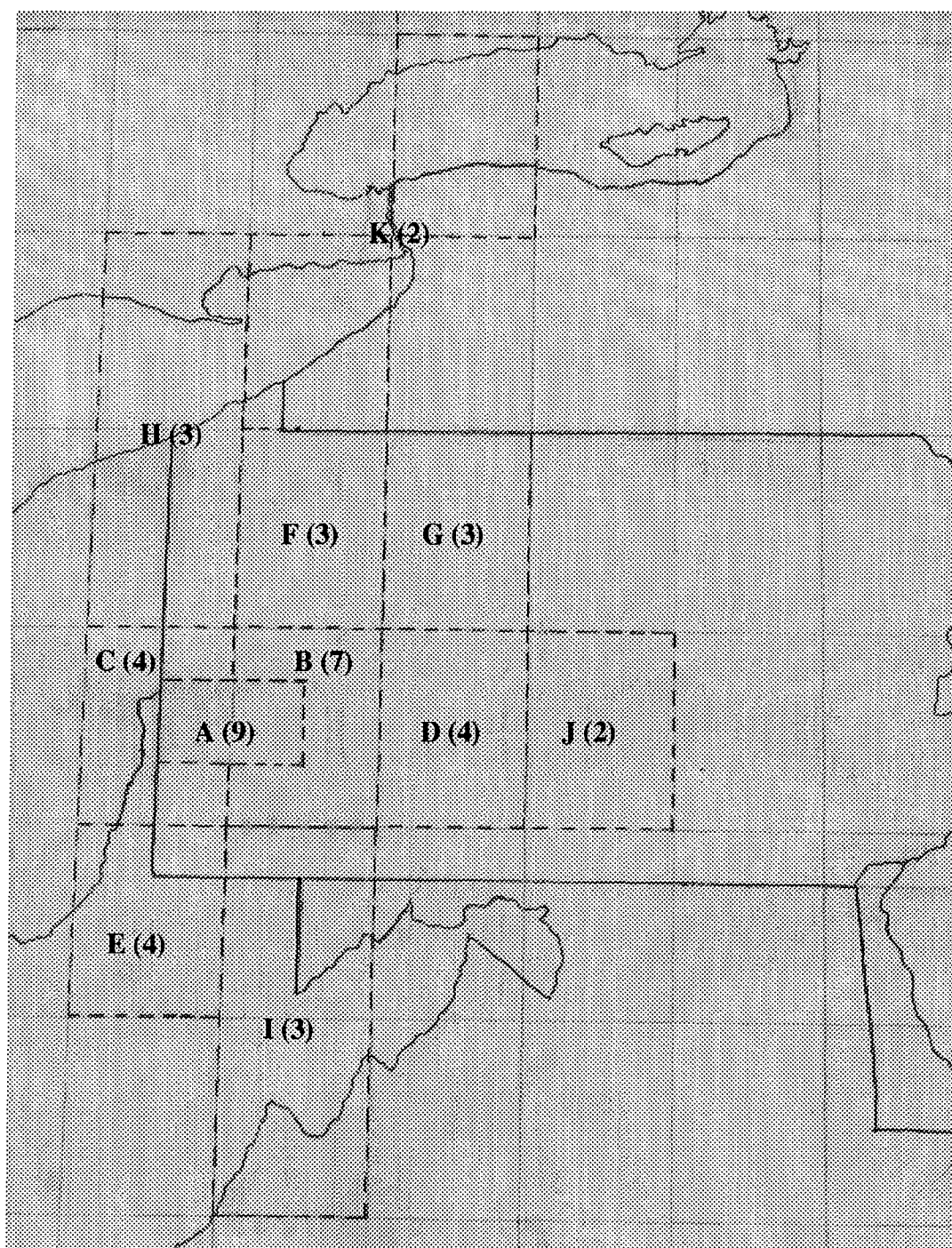
Fig. 2.6. Zones used in OPTIMIZATION & SENSITIVITY scenario.

For each set of runs, tables and mosaic plots are generated to investigate how the active DCs in the solution are affected by cost changes. The most valuable information for the client appears in Table 2.2 and Table 2.3. These crosstabulations show the frequency

of each solution network and of the individual DCs in the solutions. The DC numbering scheme is in order of appearance in the SAILS™-ODS input files. DCs numbered 1-17 are existing facilities, while those numbered 18-44 are candidate sites. The solution recommended by the baseline optimization (at the design center) and its DC components are denoted by *. Note that this solution and its components are persistent in the presence of 10% and 30% variation in the cost data. However, other DCs are also fairly persistent and merit further investigation as possible alternative recommendations.

Table 2.2. Persistence of solutions and DCs in the presence of ±10% cost variations .

DC\Solution	*2-12-17	2-12-16-21	2-12	2-17	2-17-18	2-16-17-21	2-17-34	2-18	2-18-21	Sum	% of Solutions
*2	9	4	3	2	2	2	1	1	1	25	100%
*12	9	4	3							16	64%
16		4				2				6	24%
*17	9			2	2	2	1			16	64%
18					2			1	1	4	16%
21		4				2			1	7	28%
34							1			1	4%
Total	9	4	3	2	2	2	1	1	1	25\75	

Table 2.3. Persistence of solutions and DCs in the presence of ±30% cost variations.

DC \ Solution	*2-12-17	2-18-21	2-21-23	Other Solutions (single occurrences)	Sum	% of Solutions
*2	3	3	2	8	16	64%
4				6	6	24%
7				5	5	20%
*12	3			3	6	24%
16				2	2	8%
*17	3			8	11	44%
18		3		3	6	24%
21		3	2	10	15	60%
23			2	5	7	28%
25				1	1	4%
26				3	3	12%
34				3	3	12%
35				1	1	4%
37				1	1	4%
Total	3	3	2	17	25 \ 83	

2.6. RESULTS AND CONCLUSIONS

Although CIAT has not been used at PFS LC yet, they are very enthusiastic about its potential. They were also excited about the level of automation present in CIAT and its potential extensions to include investigation of other costs and response variables.

Many of their projects involve multi-billion dollar decisions, so every bit of useful information can have a tremendous impact. Particularly useful are the alternative solutions and information on solution persistence. Moreover, the excursions performed in OPTIMIZATION & SENSITIVITY can be tailored to the client. The potential savings provided in the DATA COLLECTION & ANALYSIS stage is also substantial. For each day saved in this stage, $1000 of analyst time is saved, and the client's time is saved as well.

CHAPTER 3

Now that the CIAT prototype is in place at PFS LC, there are a few obvious extensions for the research contained in this thesis and CIAT itself. The first extensions will occur at PFS LC by the analysts and computer support people they employ. It involves duplicating and modifying Excel macros and Turbo Pascal routines so that other types of costs and responses can be incorporated in CIAT applications.

The second area for extension may also occur through use of CIAT at PFS LC. This entails developing general guidelines for grouping of cost factors in CIAT applications. Some grouping patterns will prove more efficient in that they tend to require fewer runs in order to isolate cost drivers. However, the best way to identify these general grouping guidelines is through empirical observation.

The final recommended extension is more theoretical in nature. It involves investigating or developing experimental designs most suited for use in the OPTIMIZATION & SENSITIVITY stage. The best design for application in this area would investigate the greatest number of independent vectors from the design center in a minimal number of runs. It might also be preferable to focus entirely on vectors where no costs are decreased, since the decision maker is usually most concerned with how combinations of cost increases may affect the network solution.

APPENDIX A: PROBLEM ENVIRONMENT

Supply chain management describes the process of procuring the necessary raw materials, producing products, and distributing them to customer zones. There is an implicit network configuration to this process, as demonstrated in Figure A.1. The objective of supply chain management is to design and utilize such networks at minimum cost.
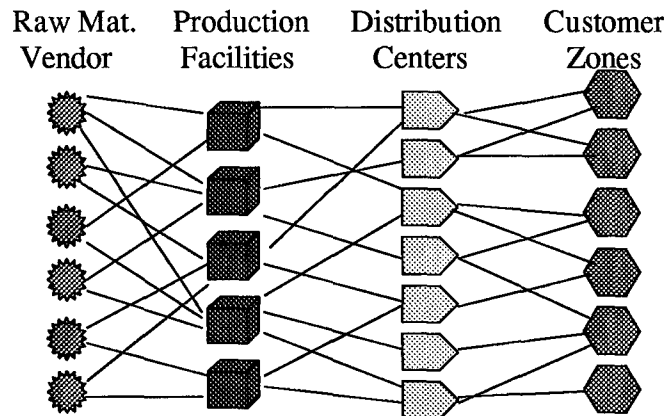
Fig. A.1. A simple supply chain.

Supply chain issues fall into three categories based upon their scope and time-frame. Operational issues are those dealing with the details of day-to-day operations, such as assigning a specific vehicle to a list of deliveries. Tactical issues are short-range plans, usually with a time horizon of less than one year, such as choosing production schedules to meet expected demands. Strategic issues are long-range plans for the supply chain, particularly decisions on the structure of the chain. (Barlaz, 1996; *Concepts: A Handbook for SAILS™ Users*, Volume 1, 1995:1-4; Ganeshan and Harrison, 1995:2)

Supply chain optimization uses mathematical programming techniques to answer the questions in strategic supply chain management:

- "How many distribution centers (DCs) are in the best (optimal) solution?",

- "Where should we place our DCs?",

- "Which DCs should each production facility feed?",

- "Which customer zones should be supplied by which DCs, and to what degree?",

and sometimes

- "How many plants are in the optimal solution?",

- "Where should the plants be located?",

- "How should products be allocated to plants?", and

- "Where should raw materials be acquired?".

The mathematical programs formulated to answer these questions are large-scale mixed integer linear programs (MIPs) with specialized structure. There is such widespread interest in supply chain optimization in today's corporations that specialized software packages have been developed to solve these MIPs. In fact, consulting agencies have developed an expertise in supply chain analysis and proper use of these optimization packages. PFS Logistics Consulting, A Division of PepsiCo, Inc., (PFS LC) is one of these agencies.

PFS management recognized their company needed this type of analysis early in the development of supply chain optimization. As a result, analysts at PFS began using a package called SAILS™ (Strategic Analysis of Integrated Logistics Systems) while it was still in development, and PFS LC was born. Today, analysts at PFS LC periodically evaluate distribution structures within their organization to find areas for improved efficiency. They also solve problems for external clients who are contemplating a change

in their supply chain or who simply want to evaluate their current way of doing business.

(Barlaz, 1996; Clarkson, 1996; Hansen, 1996; Villareal,1996)

Whatever the project, PFS LC performs the following six stages of analysis, with approximate time required given parenthetically (Hansen, 1996):

Stage 1. MODEL DESIGN (2-3 days)

Stage 2. DATA COLLECTION & ANALYSIS (1-2.5 months)

Stage 3. MODEL BUILDING (2 weeks)

Stage 4. VALIDATION (1 week)

Stage 5. OPTIMIZATION & SENSITIVITY

Stage 6. RECOMMENDATIONS   (3-5 weeks for stages 5 and 6 combined)

Stage 2, DATA COLLECTION & ANALYSIS, takes the largest portion of the project's total time. This is not surprising since the data, cost factors and location information drive the optimization results.

Based upon conversations with analysts at PFS LC, cost data used in the model is particularly time consuming to acquire and subject to error. Since some sites which are to be evaluated do not even exist at the time of the analysis, these costs are "engineered" from a zero base or by use of regression analysis of costs as a function of facility size. Even with sites already in operation, it may be difficult to properly categorize costs as fixed or variable with the size of the facility, due to inconsistant accounting procedures. (Barlaz, 1996; Clarkson, 1996; Hansen, 1996; Villareal, 1996) Clients plan to implement changes recommended in Stage 6 with potentially significant impacts on their bottom line, so they naturally want all data to be very accurate. Also, if the initial recommendations

A-3

seem particularly counter-intuitive or unacceptable to the client, the process returns to MODEL DESIGN and DATA COLLECTION & ANALYSIS to isolate and correct inconsistencies. This may improve the results of the optimization or provide a stronger base of support for the previous results. (Barlaz, 1996; Clarkson, 1996; Hansen, 1996; Villareal, 1996) Consequently, those in charge at PFS LC see an opportunity for process improvement in these six project stages. Specifically, they would like to pare down the time required for Stage 2 and perhaps begin to communicate preliminary findings to their clients in this stage in order to correct potential inconsistencies earlier in the process. (Barlaz, 1996; Clarkson, 1996; Hansen, 1996; Villareal, 1996)

The objective of this research is to assist them in their efforts to pare down the DATA COLLECTION & ANALYSIS stage and enhance client involvement and satisfaction. Since initial estimates of costs, demands, and distances used in the model are often readily available, an initial model run could use these values for a rough cut solution. Then use of a technique which identifies the parameters with the greatest impact on the solution would allow analysts at PFS LC to focus time and effort in Stage 2 on these significant parameters. Essentially what they need is a tool and procedure for performing optimality analysis on the parameters in these large-scale MIPs.

As a result of these practical issues – in addition to several theoretical considerations which make optimality analysis of objective function parameters more straightforward than sensitivity of technological coefficients and right-hand-side vectors – this research will focus on the optimality analysis of the cost coefficients.

APPENDIX B: LITERATURE REVIEW

B.1. SUPPLY CHAIN OPTIMIZATION

Although the term supply chain management had not yet been coined in 1974, in an article published that year, Geoffrion and Graves present a now standard formulation for the basic strategic supply chain optimization problem and explain a decomposition technique which solves these large-scale mixed integer programs with relative efficiency. Below, I discuss their article in great detail since the model formulation and solution technique presented by Geoffrion and Graves are the foundation of the SAILS™-ODS solver engine, one of two in the software package used at PFS LC.

One major assumption in Geoffrion and Graves' formulation makes the solution technique possible – they require that each customer zone is supplied by a single distribution center (DC). (They assert that this requirement often holds in practice.) Consequently, if sole-sourcing is inappropriate, this formulation and solution approach should not be used. (In these cases analysts at PFS LC would employ the SAILS™-OPTIMA solver, which does not make this restriction.)

Before presenting their formulation, the following parameters and variables must be defined.

Indices:	i	for commodities
	j	for plants
	k	for potential DCs
	l	for customer zones
Parameters:	S_{ij}	commodity i production capacity at plant j

D_{il} commodity i demand at customer zone l

$\underline{V}_k$, V_k lower and upper bounds on annual throughput at DC k

f_k annual fixed cost for attaining and operating DC k

v_k variable per unit cost on throughput at DC k

c_{ijkl} average per unit production and shipping cost of

          commodity i produced at plant j and sent through DC k to

          customer zone l

Variables: x_{ijkl} real-valued and non-negative amount of commodity i

          shipped from plant j through DC k to customer zone l

y_{kl} a binary variable with value 1 if DC k serves customer zone

          l, 0 otherwise

z_k a binary variable with value 1 if a DC is needed at site k, 0 if

          the solution does not include a DC at site k.

The problem formulation is then:

$$\text{Min } \sum_{ijkl} c_{ijkl} x_{ijkl} + \sum_{k} [f_k z_k + v_k \sum_{il} D_{il} y_{kl}]$$

s.t.   (1) $\sum_{kl} x_{ijkl} \leq S_{ij}, \forall_{ij}$, (Capacity Constraints)

       (2) $\sum_{j} x_{ijkl} = D_{il} y_{kl}, \forall_{ikl}$, (Demand Constraints)

                                                      (MCDS)

       (3) $\sum_{k} y_{kl} = 1, \forall_{l}$, (Sole - sourcing Constraints)

       (4) $\underline{V}_k z_k \leq \sum_{il} D_{il} y_{kl} \leq V_k z_k, \forall_{k}$, (DC Throughput Constraints)

       (5) other linear constraints on y and z as needed,

where (MCDS) stands for Multi-Commodity Distribution System. (Geoffrion and Graves, 1974:823)

An explanation of the objective function and constraints of this model is in order. The objective is to minimize the sum of production and shipping costs ($\sum_{ijkl} c_{ijkl} x_{ijkl}$), fixed costs ($\sum_k f_k z_k$), and variable costs incurred by operating the required DCs at the determined capacity ($\sum_k v_k \sum_{il} D_{il} y_{kl}$). The constraints at (1) ensure that production capacity for a given commodity is not exceeded at any plant. The constraints at (2) force demand for a given commodity to be met exactly at each customer zone. They also restrict available delivery routes to those containing a DC activated to serve the given customer zone, since y_{kl} is zero unless DC k serves customer zone l. If these constraints were formulated as a $\geq$ inequality, they would not prohibit demand from being met along routes which are closed (i.e. routes with $y_{kl} = 0$). The constraints at (3) enforce the sole-source assumption mentioned previously; they prevent any customer from being served by multiple DCs. The constraints at (4) enforce lower and upper capacity bounds on throughput at each DC, and ensure that no flow is allowed through DCs which are not opened. The linear constraints at (5) can incorporate problem-specific logical relationships in DC location combinations, sourcing of customer zones by DCs, and customer service specifications.

Geoffrion and Graves note that their (MCDS) formulation differs from previous formulations of the warehouse location problem. Earlier models use one set of variables for transportation of product to the DCs and a separate set for transportation of product from the DCs to the customer zones, instead of one set of variables with four subscripts (indexing commodities, plants, DCs and customer zones). Geoffrion and Graves assert that their formulation is more applicable to real-world instances of the problem, since it

tracks the product throughout the distribution chain. As a result, issues such as travel time constraints on perishable goods or the option to route some product directly to customers through a fictitious DC, can be accommodated. (Geoffrion and Graves, 1974:824)

Geoffrion and Graves solve (MCDS) using a variant of Benders' Decomposition. The sole-sourcing requirement allows the decomposition into a master problem in the binary variables and a standard linear transportation subproblem for each commodity. Assuming a basic understanding of Benders' Decomposition algorithm, Geoffrion and Graves' explanation of their implementation is hard to follow. Therefore, a brief review of the general Benders' Decomposition approach follows.

This explanation of Benders' Decomposition is taken from *Optimization Theory for Large Systems* by L.S. Lasdon and from a lecture by J. T. Moore. (Lasdon, 1970:370-389; Moore, 1996)

Consider the following problem which is linear in $\mathbf{x}$ for fixed $\mathbf{y}$.

$$\text{Min } \mathbf{c'x} + \mathbf{f}(\mathbf{y})$$
$$\text{s.t. } \mathbf{Ax} + \mathbf{g}(\mathbf{y}) \geq \mathbf{b} \qquad\qquad (P)$$
$$\mathbf{x} \geq \mathbf{0}, \ \mathbf{y} \in S.$$

Define the set of all feasible choices of $\mathbf{y}$ as $\mathbf{R} = \{\mathbf{y} \in S | \exists \mathbf{x} \geq \mathbf{0} \ni \mathbf{Ax} \geq \mathbf{b} - \mathbf{g}(\mathbf{y})\}$. By application of Farkas' Lemma, $\mathbf{y}$ is feasible for (P) if and only if $[\mathbf{b} - \mathbf{g}(\mathbf{y})]'\mathbf{u} \leq \mathbf{0}$, $\forall \mathbf{u} \ni \mathbf{A'u} \leq \mathbf{0}$. Since these $\mathbf{u}$ vectors form a polyhedral cone, we can write $\mathbf{u} = \sum_1^n \lambda_i \mathbf{u}_i'$, $\lambda_i \geq 0$, where the $\mathbf{u}_i'$ are the extreme rays of the cone $\mathbf{C} = \{\mathbf{u} | \mathbf{A'u} \leq \mathbf{0}, \mathbf{u} \geq \mathbf{0}\}$. So we now write $\mathbf{R} = \{\mathbf{y} \in S | [\mathbf{b} - \mathbf{g}(\mathbf{y})]'\mathbf{u}_i' \leq \mathbf{0},$

for $i = 1,2,...,n_r\}$. Thus (P) becomes $\underset{y \in R}{\text{Min}}\{f(y) + \text{Min}\{c'x | Ax \geq b - g(y), x \geq 0\}\}$. By

the duality theorem of linear programming, we can replace the inner minimization by its

dual to obtain $\underset{y \in R}{\text{Min}}\{f(y) + \text{Max}\{[b - g(y)]'u | A'u \leq c, u \geq 0\}\}$. If we let

u_i^p, for $i = 1,2,...,n_P$, denote the extreme points of the feasible region for the inner

maximization, then we have

Min z

s.t. $z \geq f(y) + [b - g(y)]'u_i^p$, for $i = 1,2,\cdots,n_P$ (MP)

$y \in R = \{y \in S | [b - g(y)]'u_i^r \leq 0,$ for $i = 1,2,\cdots,n_r\}$.

If we can find (z^*, y^*, x^*) so that (z^*, y^*) solves (MP) and x^* solves

Min $c'x$

s.t. $Ax \geq b - g(y^*)$ (SP*),

$x \geq 0$,

where the problem is denoted (SP*) to reflect its dependence upon the solution (z^*, y^*)

found for (MP), then (y^*, x^*) solves (P).

    Benders' Decomposition uses this result, and the fact that few of the constraints of

(MP) are actually binding at optimality, to define an iterative algorithm for this type of

problem. Specifically, let (MP$_0$) represent (MP) with no constraints added, i.e.

Min z

s.t. $y \in S$.   (MP$_0$)

Then use the dual to (SP$_0$)—the problem resulting when (SP*) incorporates the solution

(z^0, y^0) for (MP$_0$)—to check for optimality and add any violated constraints (called

Benders' cuts) to (MP$_0$), and repeat the process until reaching optimality. At each

B-5

iteration h, the optimality test relies upon the fact that an optimal solution (z^*, y^*) to (MP_h) is optimal for (MP) if and only if $z^* - f(y^*)$ equals the dual solution of (SP_h). The dual problem to (SP_h) is given by

$$\text{Max}_{u} \; w = [\mathbf{b} - \mathbf{g}(\mathbf{y}^h)]'\mathbf{u}$$

$$\text{s.t.} \quad \mathbf{A}'\mathbf{u} \leq \mathbf{c} \qquad\qquad\qquad (DP_h)$$

$$\mathbf{u} \geq \mathbf{0}.$$

If the solution to (MP_h) is not optimal for (MP) then construct (MP_{h+1}) by adding the constraint $[\mathbf{b} - \mathbf{g}(\mathbf{y}^h)]'\mathbf{u}^h \geq z - \mathbf{f}(\mathbf{y}^h)$, and repeat.

The algorithm is:

Step 1: Set iteration counter h to 0.

Step 2: Solve (MP_h).

If (MP_h) is infeasible, then (P) is infeasible.

Otherwise: If (MP_h) has a finite optimal solution, then denote that solution by (z^h, y^h). If (MP_h) is unbounded, then set z^h to negative infinity and select some arbitrary $\mathbf{y}^h$ in $\mathbf{S}$. Proceed to Step 3.

Step 3: Solve (DP_h).

If (DP_h) is infeasible, then (P) is unbounded.

If (DP_h) is unbounded, then go to Step 6.

Otherwise, denote the solution to (DP_h) as $\mathbf{u}^h$ and the objective function value as w^h and proceed to Step 4.

Step 4: Perform the optimality test.

If z^h - $f(y^h)$ equals w^h, then (z^h, y^h) is optimal for (MP). Find the

corresponding optimal value for **x** either by duality methods and use of $\mathbf{u}^h$,

or by solving (SP_h).

Otherwise, continue at Step 5.

Step 5: Create (MP_{h+1}) by adding the constraint $z \geq [\mathbf{b} - \mathbf{g}(\mathbf{y}^h)]'\mathbf{u}^h + \mathbf{f}(\mathbf{y}^h)$ to

(MP_h). Set h to $h+1$ and return to Step 2.

Step 6: Since the problem at Step 3 was unbounded, we need to determine the

extreme point $\mathbf{u}^h$ and extreme ray $\mathbf{v}^h$ such that the objective function value

approaches infinity along the line $\mathbf{u} = \mathbf{u}^h + \lambda\mathbf{v}^h$, for non-negative values of λ.

Then we construct (MP_{h+1}) by adding the constraint $[\mathbf{b} - \mathbf{g}(\mathbf{y}^h)]'\mathbf{v}^h \leq \mathbf{0}$ to

(MP_h). If $z^h < [\mathbf{b} - \mathbf{g}(\mathbf{y}^h)]'\mathbf{u}^h + \mathbf{f}(\mathbf{y}^h)$, then $z \geq [\mathbf{b} - \mathbf{g}(\mathbf{y}^h)]'\mathbf{u}^h + \mathbf{f}(\mathbf{y}^h)$ should

also be added to (MP_{h+1}). Set h to $h+1$ and return to Step 2.

The algorithm must conclude in a finite number of steps since we would eventually

generate all of the finite number of constraints to (MP). (Lasdon, 1970:370-389; Moore,

1996)

Returning to Geoffrion and Graves application of Benders' Decomposition to

solve their supply chain optimization problem (MCDS), in the master problem, the binary

variables are set so that constraints (3) through (5) are satisfied. The problem can then be

separated into a set of standard linear transportation problems, one for each commodity,

which are combined to form the Benders' subproblem. Solving the subproblem at each

iteration provides an upper bound on the optimal solution, since each set of transportation

networks solved is a feasible solution for the distribution system. The dual variables of the

subproblem solution are then used to add cuts to the binary master problem. Solution of the master problem at each iteration provides a lower bound on the solution, since it selects the lowest fixed and variable cost network configuration without all of the capacity and demand constraints enforced. These upper and lower bounds converge with increasing iterations, and the process stops when the difference between the least upper bound and the greatest lower bound is less than a specified tolerance on optimality. The algorithm given by Geoffrion and Graves follows. (Geoffrion and Graves, 1970:827-30)

Step 0: Set the iteration counter, H, to 0. Select convergence tolerance $\varepsilon > 0$.

Set UB to infinity and LB to negative infinity. If a binary vector solution for $\mathbf{y}$ and $\mathbf{z}$ is known which satisfies (3) through (5), then proceed to Step 2 with $(\mathbf{y}^1, \mathbf{z}^1)$ set to the known feasible solution. Otherwise, proceed to Step 1.

Step 1: Solve master problem H,

Min $Y_0 + \sum_k [f_k z_k + v_k \sum_{il} D_{il} y_{kl}]$

s.t. (3), (4), (5) from original problem formulation, and,
if $H > 0$,

$Y_0 + \sum_{ikl} \pi_{ikl}{}^h D_{il} y_{kl} \geq -\sum_{ij} v_{ij}{}^h S_{ij}$, for $h = 1,2,\cdots,H$

where Y_0 is unrestricted in sign and continuous, and π^h and v^h are the dual variable values for the demand constraints and supply constraints of linear subproblem h.

Step 2: Denote the optimal solution found by $(\mathbf{y}^{H+1}, \mathbf{z}^{H+1}, Y_0{}^{H+1})$. Set LB to the optimal objective function value. If UB is less than or equal to LB + ε, stop with the conclusion that the current solution is ε-optimal. Otherwise proceed to Step 3.

B-8

Step 3: Solve the linear subproblem,

$$\text{Min } \sum_{ijkl} C_{ijkl} X_{ijkl}$$

s.t. (1) and (2) of the original problem formulation,

with X_{ijkl} real and non-negative, with $\mathbf{y} = \mathbf{y}^{H+1}$.

Let $\mathbf{x}^{H+1}$ denote the optimal solution and $T(\mathbf{y}^{H+1})$ the optimal objective function value. Then $U = \sum_{k}[f_k z_k + v_k \sum_{il} D_{il} y_{kl}] + T(\mathbf{y}^{H+1})$ is an upper bound on the solution to the original problem. If $U < UB$, then replace UB with U, store $(\mathbf{y}^{H+1}, \mathbf{z}^{H+1}, \mathbf{x}^{H+1})$ as the current candidate solution, and stop if UB is less than or equal to $LB + \varepsilon$, with the current solution as ε-optimal. Otherwise find the dual solution for this subproblem with $\mathbf{y} = \mathbf{y}^{H+1}$, and denote it by $(\upsilon^{H+1}, \pi^{H+1})$, where υ^{H+1} corresponds to the supply constraints at (1) and π^{H+1} corresponds to the demand constraints at (2). Increment H by 1 and return to Step 1.

Geoffrion and Graves also present their practical experience with supply chain optimization and explain how lessons learned from that experience are incorporated into the formulation and solution approach presented. They outline eight types of model runs which they see as common to the supply chain optimization process.

- First are the "probationary exercises" used to test the model and begin its validation.

- Next, "regional optimization" runs are completed for further validation and to provide first insights into the distribution system solution.

- Only then is a "global optimization" run performed.

The final five types of runs address issues raised by the global optimization results and refine the information gained before presentation to decision makers.

- "What if...?" scenarios may be questions triggered by the global optimization or issues of particular interest to the decision makers.

- "Sensitivity analysis" runs vary the input parameter values to check on their impact on the optimal solution.

- "Continuity analysis" runs vary data in the constraints to see when small changes produce large differences in the solution due to the discrete nature of the solution space.

- "Tradeoff analysis" runs consider objectives which may compete with the minimization of cost, such as customer service.

- Finally, "priority analysis" runs are used to prioritize recommendations in order of decreasing savings potential.

Geoffrion and Graves explain that the Benders' Decomposition solution technique is particularly efficient for such a sequence of related runs. The Benders' cuts generated for the original problem can be adjusted to hold for small variations of the problem. With these modified cuts already in place, we solve the new problem, often resulting in a faster solution than without the modified cuts included. (Geoffrion and Graves, 1970:831, 833-836)

Note that sensitivity and continuity analysis are precisely the information required by PFS LC. However, they would like a systematic approach to this analysis which can be carried out early in the analysis process and does not involve running the model for every possible combination and level of parameter change. For the size problem dealt with at PFS LC, it is not feasible to run the tremendous number of scenarios this would involve.

B-10

Also of particular significance to my thesis research, Geoffrion and Graves confine continuity analysis to changes in the parameters of the constraints since "[c]hanges in data appearing only in the objective function...cannot lead to such behavior" as a discontinuous change in the feasible region of the problem. (Geoffrion and Graves, 1970:834) Thus, the cost coefficients should be more amenable to the type of systematic sensitivity analysis we seek than would other model parameters.

Geoffrion and Graves apply their formulation and solution approach to a large real-world problem instance for Hunt-Wesson Foods, Inc. In their implementation, they chose not to optimize the master problem at each iteration, but to stop with a feasible solution more than ε below that currently designated as UB. With this modification of the approach, the lower bound is not needed, and the stopping criterion is now met when no feasible solution to the master problem gives a value more than ε below the current UB. This sub-optimization method saves computations, particularly in the early iterations when we are presumably far from our eventual solution. They were struck by the low number of iterations required to achieve even tight tolerances, generally 3-7 iterations. They note that re-optimization for the eight common types of scenarios did not generally reduce the number of iterations required, but did provide computational savings. This real-world application for Hunt-Wesson, Inc. led them to eventual development of SAILS™-ODS, as described in the Geoffrion and Powers article. (Geoffrion and Powers, 1993)

It is interesting to note that Geoffrion and Powers see their SAILS™-ODS package as the only commercial solver which employs Benders' Decomposition, in spite of the advantages to this approach when the sole-sourcing assumption is valid. In fact,

B-11

Geoffrion and Powers are surprised and disappointed to report that heuristic approaches to supply chain design and improvement are still more widely-used than optimization approaches. (Geoffrion and Powers, 1993:9-10)

R.L. Breitman and J.M. Lucas outline a tool used by General Motors for all aspects of supply chain management. (Breitman and Lucas, 1987) Their system, PLANETS, is a flexible system designed to model a wide array of management decisions made at all levels of GM. PLANETS was developed in 1974 to enable GM decision makers to formulate and solve mathematical models addressing recurring questions and ad hoc analysis issues. The current network of computer programs has a user interface which is menu-driven and uses conversational business terminology rather than mathematical programming lingo. Once a business situation of interest is outlined through use of the user-interface, PLANETS automatically formulates and solves the associated mixed-integer programming problem, using MPSX and SCICONIC as the solver engines. Reports are generated according to the users selections from such groups as "process center utilization", "material usage", "product cost allocation", "product-shipment-by-facility", "investment summary", "facility-revenue-and-cost", and "facility-income-and-cash-flow". Reports on the sources of infeasibility and sensitivity to parameters are also available. (Breitman and Lucas, 1987:97-98)

Breitman and Lucas observe that a typical PLANETS problem formulation has over 10,000 variables, over 100 of which are restricted to integer values. Yet, PLANETS formulates the problems and generates structured input for the solver in about one to two minutes. (Breitman and Lucas, 1987:97)

It is not clear how the sensitivity analysis report is generated for such large-scale mixed integer programming problems. Breitman and Lucas do not divulge the methodology used to generate this report. In fact, they do not even state whether this report is built by PLANETS or is part of the output provided by the MPSX or SCICONIC solver. Moreover, we do not know if the sensitivity report provides information on how much simultaneous change can occur in the parameters, or if parameters are tested only one at a time.

B.2. OPTIMALITY ANALYSIS IN LINEAR AND INTEGER PROGRAMMING

Once we solve a mathematical programming problem, our concern turns to how small changes in the problem data affect the solution. This need arises because the model's parameter values are often mere estimates or postulates of the true but unknown values. Consequently, widespread investigation has produced a plethora of theory and techniques on optimality analysis in linear programming.

For example, we can substitute $\mathbf{b} + \theta\mathbf{v}$ for the right-hand-side vector $\mathbf{b}$, where $\mathbf{v}$ represents the direction of change and θ is a nonnegative scalar. Starting with the original optimal solution (at $\theta = 0$) we increase θ until loss of feasibility for the optimal basis at some $\theta = \theta_1 > 0$. Then the dual simplex method is used to find the new optimal solution which comes into effect at θ_1. The process repeats to find the remaining breakpoints θ_2, θ_3, ..., θ_N until further increases in θ have no effect; i.e. the basis remains optimal for additional changes in θ or no feasible solution exists for $\theta > \theta_N$. (The cost vector can similarly be parametrized and investigated.) See Bazaraa, Jarvis, and Sherali for a complete development of these parametric analysis techniques. (Bazaraa, Jarvis, and

Sherali, 1990:278-301) Also see Ward and Wendell for a unified approach to optimality

analysis that generalizes the above approaches and presents the full theoretical basis for

these methods in linear programming. (Ward and Wendell:1990)

All approaches to optimality analysis – a term encompassing sensitivity analysis,

parametric analysis, and other techniques – explained so far are rather limiting in that they

restrict parameter investigation to occur along a linear cut specified by the direction vector

v. An alternative approach applies the statistical techniques of experimental design and

response surface methodology to construct a surface mapping of the optimal objective

value as a function of the problem data.

P.W. Smith first proposed this use of design of experiments (DOE) and response

surface methodology (RSM) as an approach to optimality analysis of parameters in 1975.

His methodology describes the effect of simultaneous parameter variation in two types of

mathematical programming formulations of an economic model: a linear programming

formulation and a formulation with linear constraints and a quadratic objective function.

In all formulations the variables used were continuous in nature. (P.W. Smith, 1975)

Although the peculiarities of the economic models under investigation obscure his

general methodology, it is clear that he focuses on simultaneous but distinct changes in the

elements of the right-hand-side vector, **b**. He also uses a second-order experimental

design, containing design points required to estimate the parameters in a quadratic

response surface through least squares regression, where the factors fixed at each design

point are the elements of **b**.

Smith also recognizes four areas of concern in his approach:

B-14

1. The right-hand-side elements in his problems are interdependent, greatly restricting his freedom in design selection. Specifically, some settings of b_i precluded certain simultaneous settings of b_j. (P. W. Smith, 1975:54-55)

2. Realistic economic models contain too many coefficients for practical application of experimental design. (P. W. Smith, 1975:44)

3. The true response surface is a faceted figure with sharp corners and discontinuities rather than the smooth continuous surface constructed through use of DOE and RSM. (P. W. Smith, 1975:77)

4. Experimental designs concentrate on minimizing variance error rather than the bias error upon which his analysis focuses. (P. W. Smith, 1975:34, 41-2)

K. E. Johnson et. al. address items (2) and (3). (Johnson, Bauer, Moore, and Grant, 1996) They also give a clear and detailed outline of the approach, as summarized below.

The authors deal with the general linear programming model

Minimize $\mathbf{c}'\mathbf{x}$

subject to: $\mathbf{Ax} = \mathbf{b}$

$\mathbf{x} \geq \mathbf{0}$,

where $\mathbf{c}$ is an n-vector of specified cost coefficients, $\mathbf{A}$ is an $m \times n$ matrix of given technology coefficients, $\mathbf{b}$ is an m-vector of known resource levels called the right-hand-side vector, and $\mathbf{x}$ is an n-vector of variables. Focusing on a technique for optimality analysis of $\mathbf{b}$, their general outline of the response surface methodology procedure is:

Step 1: Specify a high and low range for each element of $\mathbf{b}$, b_i. Code these so that the high level is given by $z_i = +1$ and the low level by $z_i = -1$ by using

B-15

z_i = (b_i - midpoint)/half-range. Use experimental design techniques to specify

the combinations of parameter levels (i.e. combinations of z_i = +1 or z_i = -1 for

i = 1, 2, ..., m) to be run in the LP. Let s represent the number of runs

specified by the chosen experimental design.

Step 2: Solve the LP for each of the s parameter level combinations, or design

points, specified in the experimental design from Step 1. Record the optimal

objective function value for each of these design points as y_k.

Step 3: Perform least squares regression to estimate the coefficients, β, of the

proposed functional form, $\mathbf{y} = \mathbf{Z}\beta + \epsilon$, where $\mathbf{Z}$ is an $s \times q$ matrix whose rows

give the design points determined in Step 1 along with columns for z_i^2 and $z_i z_j$

terms and often a column of 1's to give the model an intercept term, $\mathbf{y}$ is the s-

vector of responses measured in Step 2, and ϵ is the error due to omission of

higher order terms in the functional form. The resulting estimate of β is given

by $\hat{\beta} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$.

The estimate of β_i gives the magnitude and direction of a change in the optimal objective

function value with a one unit change in z_i. In the uncoded variable space, we can see that

β_i gives the magnitude and direction of a change in the optimal objective function value

with a change equivalent to the half-range specified for b_i.

The authors suggest extending their technique to provide accurate predictions of

the optimal objective function value for the LP with any specified levels of $\mathbf{b}$ within the

ranges explored. Normally, predictions of the response for given values of the predictors

would be obtained from a fitted regression model as $\hat{y} = \mathbf{z}_p' \mathbf{b}$, where $\mathbf{z}_p$ gives the coded

values of the specified settings for the predictors. Recall that all of the error in ε was due to lack of model fit. Thus, it is reasonable to believe that we can calculate an estimate for the lack of fit error at z_p, and add this to $\hat{y}$ in order to obtain a more accurate prediction.

The authors propose using a geostatistical technique called *kriging* for estimating this lack of fit error at z_p, denoted $\hat{\varepsilon}_p$. Kriging essentially uses a weighted average of the prediction errors at the design points in order to estimate $\hat{\varepsilon}_p$, so $\hat{\hat{\varepsilon}}_p = \mathbf{w}'\hat{\varepsilon}$, where $\hat{\varepsilon}$ is the vector of residuals at the s design points and $\mathbf{w}$ contains the selected weights. The errors associated with design points near z_p are weighted more heavily than those far from z_p.

Optimal weights are chosen by solving $\begin{bmatrix} \gamma & 1 \\ \hline \mathbf{1}' & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{w} \\ \hline \lambda \end{bmatrix} = \begin{bmatrix} \gamma_p \\ \hline 1 \end{bmatrix}$, where γ is an $s \times s$ matrix with

$(\gamma)_{kl}$ equal to the covariance between the kth and lth design points, $\mathbf{1}$ is an s-vector of ones, λ is a Lagrangian multiplier, and γ_p is an s-vector with $(\gamma_p)_k$ equal to the covariance between z_p and design point k. (Johnson, Bauer, Moore, and Grant, 1996:53)

Responding to the large number of potential factors in a comprehensive economic model, Johnson et. al. propose using factor screening design techniques in the initial stages of analysis on a model with a large number of factors. Once the most significant factors are identified, the rest of the process continues as outlined above with these important factors. (Johnson, Bauer, Moore, and Grant, 1996:52)

Although discontinuity and lack of smoothness noted by Smith remain a concern, kriging lessens the impact on predictions made from the response surface metamodel (adding back some constant estimate of misspecification error would not address this concern). Since kriging gives more influence to errors measured near the point of

B-17

prediction, it should adjust the response surface metamodel in areas where the smoothing is most artificial.

Although the concerns expressed about linear programming parameters are also inherent in the realm of integer programming, the methods used in continuous linear programming cannot be directly extended to integer linear programs. Since integer programming does not have the same continuity and duality properties of continuous linear programming, explanations of optimality analysis theory and techniques are scarce for integer programming. Responding to this need, A.M. Geoffrion and R. Nauss present the limited theoretical basis for sensitivity analysis in integer programming. (Geoffrion and Nauss:1977) Since no unified approach to sensitivity analysis in this realm exists for application to all types of problem parameters, Geoffrion and Nauss define two situations of interest:

$$\underset{x \geq 0}{\text{Min}}(c + \theta \ f)'x$$

s.t. $Ax \geq b$

$\qquad x_j \geq 0$ and integer $\forall j$,

$\qquad 0 \leq \theta \leq 1$,

$\hspace{8cm}(P^\theta)$

and

$$\underset{x \geq 0}{\text{Min}} \, c'x$$

s.t. $Ax \geq b + \theta \ r$

$\qquad x_j \geq 0$ and integer $\forall j$,

$\qquad 0 \leq \theta \leq 1$.

$\hspace{8cm}(P_\theta)$

(Geoffrion and Nauss, 1977:453)

In (P_θ), changes in the constraints change the feasible region of the problem, often causing a discontinuous "jump" in the solution. However, in (P^θ), changes in the objective function parameters have no effect on the problem's feasible region. According to Geoffrion and Nauss' Propositon 3.3, "[t]he optimal value of (P^θ) for $0 \le \theta \le 1$ is piecewise-linear, continuous, and concave on its finite domain." (Geoffrion and Nauss, 1977:459) Denoting this optimal value function for (P^θ) as $z(\theta)$, this proposition implies that parametric analysis of the objective function coefficients may proceed in the same fashion for integer programming as in linear programming. However, the procedure for determining $z(\theta)$ is not as convenient now that pivots of a simplex tableau are no longer sufficient to determine its breakpoints.

In an integer programming context where solution approaches are not continuous in nature, we must solve the problem with fixed values of θ. Geoffrion and Nauss suggest a procedure for selecting values of θ which most efficiently determine the breakpoints of $z(\theta)$. They denote the upper and lower bounds of the region of uncertainty about $z(\theta)$ as $UB(\theta)$ and $LB(\theta)$. $UB(\theta)$ is given by the "lower envelope" of $\mathbf{c'} \mathbf{x}^k + \theta\mathbf{f'} \mathbf{x}^k$ where $\mathbf{x}^k$ are known feasible solutions to (P^θ). $LB(\theta)$ is found through linear interpolation between optimal values solved for given values of θ. (Geoffrion and Nauss, 1977:459-460)

We can see this in Figures B.1 and B.2. Figure B.1 represents the beginning stages of the parametric analysis. The problem has been solved to optimality for $\theta = 0$ and $\theta = 1$, and the optimal objective function values for these solutions are shown at the points A and B, respectively. Cuts forming the current upper bound, $UB(\theta)$, are found by projecting the line $(\mathbf{c} + \theta\mathbf{f})' \mathbf{x}^*$ where $\mathbf{x}^{*i}$ is the optimal solution found at $i = A$ or $i = B$. This holds

B-19

since in a minimization problem the only reason the optimal solution would change along these cuts – i.e., as θ ranges from 0 at A or 1 at B – is if a lower objective function value was found with some solution other than x^{*i}. The region of uncertainty surrounding $z(\theta)$ is shaded. Note that known feasible solutions for some fixed value of θ can be used similarly to add more cuts on $UB(\theta)$ and reduce the region of uncertainty.
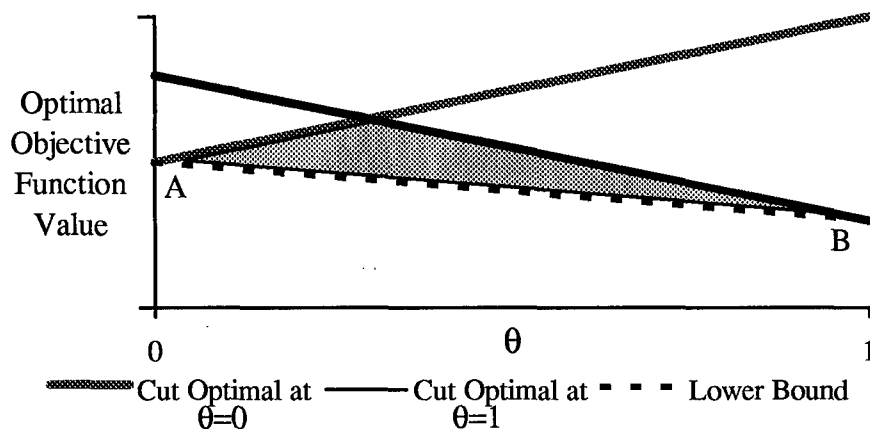
Fig. B.1. Early stages of parametric analysis.

Figure B.2 shows how the region of uncertainty is reduced as optimal solutions are found for values of θ in the interval (0,1). Each solution found is used to add a cut to $UB(\theta)$ and a new point for $LB(\theta)$, thus eliminating part of the region of uncertainty. Note how markedly the region is reduced by adding the cut which is optimal at $\theta=0.5$ and defining the new lower bound envelope as the segments connecting A and B to the optimal objective function value at $\theta=0.5$ which is labeled C. Therefore, in order to most rapidly eliminate the area of uncertainty about the optimal value function, we should always select the next setting of θ so that $UB(\theta)$ - $LB(\theta)$ is greatest. (Geoffrion and Nauss, 1977:460) For bounded problems, the piece-wise linear optimal value function

B-20

$z(\theta)$ for (P^θ) contains a finite number of segments, and can be determined by solving for a finite number of values of θ.
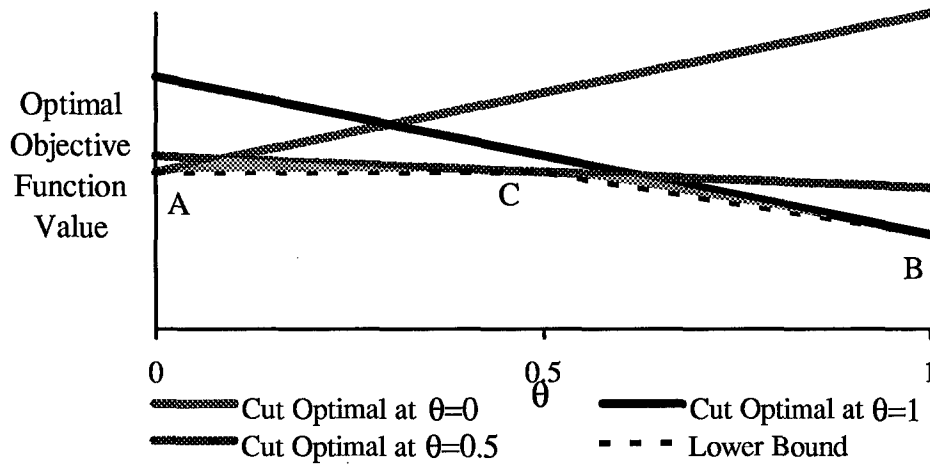
Fig. B.2. Reducing the region of uncertainty in a parametric analysis.

Geoffrion and Nauss also present techniques of Lagrangian relaxation and other methods for obtaining additional lower bounds on $z(\theta)$. They give a corollary which allows these bounds to be incorporated in the parametric analysis process outlined above. The corollary states that the upper concave envelope formed by combining $LB(\theta)$ from the above process and some other known lower bound function of θ is another valid lower bound. (Geoffrion and Nauss, 1977:461)

B.3. EXPERIMENTAL DESIGN AND RESPONSE SURFACE METHODOLOGY

The purpose of design of experiments (DOE) is to allow the experimenter to adequately illuminate his areas of interest in an economical fashion, with a design containing experimental runs so that just enough data is collected to test the appropriate hypotheses. Response surface methodology (RSM) incorporates designed experiments and least squares regression to fit an appropriate functional form to the data collected.

B-21

RSM generally begins with first-order designs and their accompanying models, used to identify the most significant factors and an appropriate region for further investigation. The second phase of RSM uses second-order experimental designs to fit quadratic polynomials to the data, providing a more detailed picture of the response surface. The purpose of RSM may be to illuminate the functional relationship between the factors set in the experiment and the response measured. RSM may also be used to identify factor settings which can be used to optimize the response, or achieve some pre-specified levels of one or multiple responses. (Myers and Montgomery, 1995:12) See Myers and Montgomery for an introduction to RSM.

A small example of a first-order experimental design on two factors is shown at Figure B.3.

Intercept	A	B
+ 1	- 1	- 1
+ 1	+ 1	- 1
+ 1	- 1	+ 1
+ 1	+ 1	+ 1

Fig. B.3. A first-order experimental design on two factors.

Denoting the design matrix as D, each row of D represents a design point to be run in the experiment, each column gives the specified levels for its respective factor in those design points. The intercept column is included to provide a constant term when a regression model is fit to this data. It does not specify factor settings. The experiment shown in Figure B.3 is a two-level factorial experiment, since each factor is set at two levels, -1 for the low level and +1 for the high level. (The experiment is called factorial since including all combinations of factor levels causes the number of runs to increase as

$n = L^k$, where L is the number of levels and k is the number of factors.) Two-level

factorial experiments are used for investigating the effects of individual factors and factor

interactions such as AB on the response, y. Thus, a factorial experiment allows estimation

of the parameters in a functional form such as $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2$, where x_1

corresponds to coded values of factor A and x_2 corresponds to coded values of factor B.

The parameter estimates found through least squares regression are given by

$\beta = (D'D)^{-1}D'y$.

The most important property of a first-order design is *orthogonality*. An

orthogonal design permits independent estimation of parameters in the response surface

functional form; i.e., each β_i is estimated so that it remains unchanged with the addition or

deletion of other terms in the regression model. Note that the dot-product of any two

columns in Figure B.3 is zero, thus this design is first-order orthogonal. While the

definition of an orthogonal design matrix changes slightly for higher-order designs, its

benefits remain the same.

After the proper region of investigation has been identified and the pool of

potentially important factors has been screened through the use of first-order designs,

center point runs are added to the design. This augmented design allows for testing the

presence of pure quadratic terms such as x_1^2 and x_2^2. The presence of these terms signals

that it is time to proceed to the second-order phase of RSM.

When a quadratic functional form is desired or indicated by the hypothesis test on

a first-order experiment with center points, we must add more design points. The most

common second-order design is called the central composite design (CCD), which is

constructed by adding axial points and an appropriate number of center points to a standard two-level factorial design. The axial points are generally placed at a distance $\alpha = \sqrt[4]{n_f}$ from the center, where n_f is the number of design points in the factorial core; since this ensures that the variance in a prediction made from the estimated response surface depends only upon the distance of that prediction point from the design center (a property called *rotatability*). Rotatability is generally more valuable than orthogonality at this stage of RSM. Thus, in cases where it is not possible to build a design which is both rotatable and second-order orthogonal, we prefer a design which is rotatable and "near" orthogonal.

The second-order design is used to estimate the parameters of a second-order polynomial model such as $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2$. As in the first-order phase of RSM, statistical tests are used to determine if any of the terms in the model can be dropped to achieve a simpler form. This second-order model can be viewed as a second-order Taylor polynomial approximation for the true underlying function and used to make predictions and provide insights about the system.

Once we get past the basics of the first- and second-order phases of RSM, we see that the details of these two phases have also been rich mines for research. In the realm of first-order experimental design, there has been a wealth of study on ways to reduce the tremendous number of experimental runs required by a factorial experiment. If a large number of factors, k, are potentially important, we can no longer afford the 2^k runs required for a two-level factorial experiment.

One approach to the problem is the use of fractional factorials (2^{k-p} designs), which require only a $1/2^p$ fraction of the design points specified in a full factorial experiment on k factors. In such cases, the design columns representing levels of individual factors will no longer be orthogonal to design columns representing their interactions, thus confounding their effects. If two factors are confounded, then we are not able to separate the effects on the response contributed by these two factors. However, with careful selection of confounding patterns, fractional factorial designs can provide a very efficient approach to screening for main effects and possibly for low-level interactions as well.

Intercept	A	B	C	D = A B C
+ 1	- 1	- 1	- 1	- 1
+ 1	+ 1	- 1	- 1	+ 1
+ 1	- 1	+ 1	- 1	+ 1
+ 1	+ 1	+ 1	- 1	- 1
+ 1	- 1	- 1	+ 1	+ 1
+ 1	+ 1	- 1	+ 1	- 1
+ 1	- 1	+ 1	+ 1	- 1
+ 1	+ 1	+ 1	+ 1	+ 1

Fig. B.4. The 2_V^{4-1} design.

Figure B.4 shows the 2_V^{4-1} design. The Roman numeral given as a subscript is the resolution of the design. Design resolution indicates the level of confounding present in the design. For example, in a resolution III design, main effects are confounded with two-factor and higher-order interactions but not with other main effects. In a resolution IV design, main effects are not confounded with other main effects or with two-factor interactions, but two-factor interactions are confounded with each other and all higher-order interactions are confounded. Resolution V, as shown in Figure B.4, provides main effect estimation which is not confounded with any interaction lower in degree than a

four-factor interaction. Also, two-factor interactions and three-factor interactions are confounded with each other.

A related group of first-order experimental designs are the Plackett-Burman designs. Plackett-Burman designs allow k factors to be investigated in as few as $k + 1$ experimental runs, if $n_{PB} = k + 1$ is a multiple of four. Otherwise, we would require n_{PB} runs, where n_{PB} is the smallest multiple of four greater than k. If n_{PB} runs are affordable, we should use a Plackett-Burman screening design, due to the acceptable confounding patterns they exhibit. However, sometimes we cannot afford even k experimental runs to investigate k factors.

Supersaturated experimental designs are those which investigate k factors in fewer than k runs. Research in this area has focused almost exclusively on three types of supersaturated designs and their application to sensitivity analysis of large computer simulation models. These three types are random balance designs, systematic supersaturated designs, and group-screening designs. (Kleijnen, 1987a; Kleijnen, 1987b; Kleijnen, 1975; Mauro, 1986; Mauro, 1983; Mauro and Burns, 1984; D. E. Smith and Mauro, 1982)

Several statisticians have reviewed supersaturated designs and noted the pros and cons of each type. C. A. Mauro and J. P. C. Kleijnen have been prolific in this area of research. Their works provide the information in the following paragraphs.

Although the definition of random balance designs first proposed by Satterthwaite in 1959 is very broad, "random balance" is now used to denote a very specific type of coded design matrix. (Satterthwaite, 1959:112-113) These random balance designs are

B-26

built by specifying an even number of runs, N, for the experiment and the number of factors to be investigated, k. The design matrix D is then $N \times (k+1)$. The -1 and +1 entries of D are assigned by random sampling without replacement so that each column contains N/2 negative one entries and N/2 positive one entries. (Satterthwaite, 1959:113) Advantages to this approach are that N is controlled and the design is simple to construct for any even value of N. (D. E. Smith and Mauro, 1982:51; Mauro, 1986:4-5) There are two significant disadvantages to random balance designs, however. First, D cannot be orthogonal for $N < k+1$, and for random balance designs the confounding pattern is random. (D. E. Smith and Mauro, 1982:51; Mauro, 1986: 5) Second, since D has fewer rows than columns, the estimators of β_i are biased. (Kleijnen, 1987a:2889; Kleijnen, 1987b:32; Mauro, 1986:7) Another source of concern is the lack of specialized analysis techniques for the results of a random balance design. Generally, factors are deemed significant if their parameters pass an individual F-test. (D. E. Smith and Mauro, 1982:51; Mauro, 1986:5)

Systematic supersaturated designs are the results of attempts to remedy the problems of random balance designs. They were developed by Booth and Cox in 1962. They build D so that it is as nearly orthogonal as possible. This is achieved by minimizing the largest pair-wise dot product of columns of D. Booth and Cox provided a computer algorithm which would construct the systematic supersaturated design for combinations of N and k with $N < k \leq 36$. However, for large k, it may not be practical to run the computer algorithm, and for $2N < k$, systematic supersaturated designs may not be much better than random balance. (Booth and Cox, 1962; D.E. Smith and Mauro, 1982:51)

Group-screening is an entirely different approach. In group-screening

experiments, the k factors of interest are subdivided into g groups of factors, G_1, G_2, ...,

G_g. In the first stage of experimentation, the factors within a group are treated as if they

were a single factor, so our design matrix, D, contains only g + 1 columns. Generally, a

resolution IV fractional factorial or Plackett-Burman design is selected for the grouped

design. (Kleijnen, 1975:492) For each experimental run with a -1 in the G_i column, all

factors in group G_i are set to their low level. Similarly, where a +1 appears under G_i, all

factors in group G_i are set to their high level. The experimental results are analyzed in the

standard fashion for the fractional factorial or Plackett-Burman design selected, and

factors are dropped from consideration if their group is found to have a statistically

insignificant effect. The process repeats itself with smaller groups until all factors have

been ruled out or tested individually. (Kleijnen, 1987a; Kleijnen, 1987b; Kleijnen, 1975;

Mauro, 1986; Mauro, 1983; D. E. Smith and Mauro, 1982)

At any stage in group-screening, the factors within a group are completely

confounded with each other. Thus, it is important that factors with opposite effects, e.g.

$+\Delta$ and $-\Delta$, be placed in different groups. Otherwise, significant factors may be dropped

from consideration due to cancellation within their group. However, if factors with

significant interactions are placed within the same group, and if a resolution IV design is

used, there should be no confounding of factor main effects between groups. (Kleijnen,

1975:490)

The advantages of group-screening are the control exerted over confounding and

the ease of application and analysis. The disadvantages are the requirement for a priori

knowledge about the effects of factors and the lack of control over the total number of runs required. (Kleijnen, 1987a; Kleijnen, 1987b; Kleijnen, 1975; Mauro, 1986; Mauro, 1983; D. E. Smith and Mauro, 1982)

Most practitioners seem to favor the group-screening approach over random balance and systematic supersaturated methods. In many practical instances, it seems that the prior knowledge required by group-screening is available. Moreover, the number of runs may be further reduced by placing factors which we expect to be significant in small groups and factors expected to be insignificant in large groups.

Once the factor-screening is completed in the first-order stage of RSM, a second-order experiment is employed and a second-order model is fitted to the data. Thus, at this stage in the process we become more concerned with bias error due to model misspecification. The concern with bias is particularly important in the case of a deterministic experiment such as a mathematical program, where experimental error does not exist.

B.K. Ishihara compares bias minimizing and variance minimizing experimental designs used with deterministic models as the experiment. (Ishihara, 1985) Central composite designs and other second-order rotatable designs often employed in RSM are used as the variance minimizing designs. The bias minimizing designs used are more obscure, since practitioners traditionally concentrate more on the variance. Both types of experimental design are applied to the Arsenal Exchange Model, a linear programming model to which RSM had been previously applied at the Air Force Institute of Technology. (Ishihara, 1985:1-7 to 1-8) Ishihara defines effectiveness as the sum of

squared deviations between the response surface prediction and the actual value for a random selection of points, with efficiency measured by the number of design points required. (Ishihara, 1985:1-8 to 1-9)

Ishihara found no significant differences between the variance minimizing and bias minimizing designs with respect to the predictive ability and the explanatory power they provide. He notes that the effectiveness and efficiency goals are competing objectives in the designs studied. However, he found that the design properties of orthogonality and rotatability are both important in this area. Since many widely used designs provide "near" orthogonality or "near" rotatability, he recommends further research to delineate these gray areas and outline levels of "near" orthogonality and "near" rotatability which provide adequate results. (Ishihara, 1985:4-2 to 4-4)

APPENDIX C: THE SAILS™ SYSTEM

C.1. OVERVIEW OF SAILS™

SAILS™ is the supply chain optimization software system produced by INSIGHT, Inc. A standard in the industry, this is the system used at PFS LC. It includes user interfaces for model building, run launching, solution review, and presentation along with two underlying mixed integer linear programming solver engines (ODS and OPTIMA). The interfaces for data input and model building are pop-up menus and spreadsheet screens which generate a collection of formatted ASCII files with various extensions for use by the solvers. The input files can be modified through the user interface or directly as long as their format is retained. PFS LC analysts use the interfaces to build base models for each project. However, when minor modifications are made, experienced analysts find it more expeditious to change the ASCII files directly.

Once a model is launched and the solver successfully completes its iterations, a second collection of formatted ASCII files are generated as solution reports. These reports can be viewed directly or displayed graphically by SAILS™ solution display interfaces. The set format of the output reports allow experienced analysts to quickly find the data elements they are most interested in. The format also allows creation of programs to find certain data in repeated applications.

C.2. SAILS™ INPUT DATA FILES

The input data files have the following extensions: _.cdf, _.ebf, _.ecf, _.faf, _.mcf, _.run, _.sdf, _.sgf, and _.tdf. The files contain information required for the SAILS™ solvers to successful execute, as detailed in Table C.1. below. (Deviprasad, 1996)

Table C.1. SAILS™ input files

File Type	Extension	Header Designation	Description
System Definition File	_.sdf	MATERIAL	List of Raw Materials Group
		PRODUCTS	List of Finished Products Group
		SUPPLIER	List of Raw Materials Supplier Locations
		PRODUCER	List of Plants/Vendor Locations
		WAREHOUS	List of Distribution Center Locations
		CUSTOMER	List of Customer Region Locations
		CUSTCLAS	Customer Classes
		COMPONEN	List of Component Products
		BNDLNAME	Product Bundle Names
		TIMENAME	Time Period Definitions
		ECHELONS	Network Echelons
		MATLCODE	Raw Material Recoding Data (for use with Transactions file)
		PRODCODE	Finished Product Recoding Data (for use with Transactions file)
		SUPPCODE	Raw Material Supplier Recoding Data (for use with Transactions file)
		PLNTCODE	Plant Location Recoding Data (for use with Transactions file)
		WHSECODE	Distribution Center Recoding Data (for use with Transactions file)
		TERRCODE	Customer Region Recoding Data (for use with Transactions file)
		REGNCODE	Forecast Database Cross Reference (for use with Transactions file)
		CLASCODE	Customer Class Recoding Data: Customer Class Codes (for use with Transactions file)
		CUSTCODE	Customer Class Recoding Data: Customer Account Numbers (for use with Transactions file)
Facility Data File	_.faf	SUPPCOST	Raw Material Supplier Costs and Capacities
		PLNTCPTY	Plant Location Fixed Costs and Capacities
		PLNFCTR	Plant Location - Finished Product Capacity Conversion Factors
		COMPCOST	Production Cost and Capacities: Component Products
		COMPLINE	Joint Component Production Line Capacity Constraints
		PLNTCOST	Production Cost and Capacities: Finished Products
		PLNTLINE	Joint Finished Product Production Line Capacity Constraints
		RMCPCNVT	Raw Materials --> Component Products Conversion/Blending Data
		CPFPCNVT	Component Products --> Finished Products Conversion/Blending Data
		RMFPCNVT	Raw Materials --> Finished Products Conversion/Blending Data
		PRODSTND	Finished Product Specific Data
		WHSECPTY	Distribution Center Costs and Capacities
		WHSECOST	Distribution Center Product-Specific Variable Costs
		INBMILES	Inbound Link Distances
		REPMILES	Replenishment Link Distances
		INTMILES	Transfer Link Distances
		OUTMILES	Outbound Link Distances
		MATLBLND	Raw Material Viscosity Indices
		COMPBLND	Component Product Viscosity Indices
Transport Data File	_.tdf	IBWTDRAT	Inbound Rates from Suppliers to Plants
		IPWTDRAT	Rates for Interplant (Intraplant) Transfer of Component Products
		RPWTDRAT	Replenishment Rates from Plants to Warehouses
		TRWTDRAT	Transfer Rates between Warehouses
		OBWTDRAT	Outbound Rates from Warehouses to Customer Zones
Demand Data File	_.cdf	CUSTDMND	Demand by Customer Zone and Product Group
		TRANSACT	Transactions File
Scenario Generation File	_.sgf	SUPPELIG	Raw Materials Supplier Eligibility
		PLNTELIG	Plant Location Finished Product Production Eligibility
		PRODELIG	Finished Product Production Eligibility
		REPLINKS	Replenishment Link Selections
		INTLINKS	Transfer Link Selections
		BUNDLING	Bundling Strategy Definitions
		WHSELIG	Distribution Center Stockage Eligibility
		STCKELIG	Finished Product Stockage Eligibility
		CLASELIG	Distribution Center - Customer Class - Shipment Type Eligibility
		OUTLINKS	Outbound Link Selections
		PLNTSVCE	Plant Location (Replenishment) Service Limits
		DISTSVCE	Distribution Center (Transfer) Service Limits
		WHSESVCE	Customer Service Limits - Distribution Centers
		CLASSCAL	Customer Class Scale Factors
		RDMDSCAL	Composite Demand Scale Factors: Customer Regions
		RPSELECT	Replenishment Weighted Rate Category Assignments & Selection Options
	.ecf		Facility Status, Reports Requested
	.ebf		Binary Version of _.ecf File
	.run		Runtime Parameters

C-2

C.3. SAILS™ OUTPUT REPORT FILES

The output data files have the extensions: _.log, _.min, _.rpt, _.tbl. The _.log file contains runtime information and error codes. The _.rpt file is the full solver report on the solution obtained. The _.min file is a summary of the full solver report. The _.tbl is the final solver table, and can be deleted after each run. (Deviprasad, 1996)

APPENDIX D: CIAT USERS' GUIDE

D.1. STRUCTURE

This appendix begins with a flow chart of the CIAT application process. The flow chart is annotated with time estimates and each component is numbered for reference purposes. The flow chart is followed by a section detailing each numbered component. The flow chart and accompanying details are designed to provide a practical users' guide to CIAT.

D.2. CIAT PROCESS FLOW CHART

1 SAILS base model.

Beginning of CIAT application? — Yes / No

2 Extract costs from SAILS input files into Excel "Work Space". (approx. 5 min.)

3 Set upper and lower bounds on costs to be investigated. (approx. 1 min.)

More than 31 costs specified? — No / Yes

Use individual costs as factors.

4 Group costs; use groups as factors. (approx. 3 min.)

5 Generate design table in JMP for factors. (approx. 10 sec.)

6 Bring coded design table into "Work Space" & calculate true settings for design runs. (approx. 5 min.)

7 Create SAILS files for runs in design table. (approx. 5-10 min.)

8 Load & launch SAILS runs. (approx. 2 min.)

(There is a problem dependent time lapse here while the SAILS solver runs.)

9 Extract response values from SAILS _.min files & add to coded table in JMP. (approx. 5-10 min.)

10 Analyze model w/ linear least squares regression. (approx. 5 min.)

Was grouping used? — Yes / No

11 Drop groups which have relatively small sum of squares.

(approx. 5 min.)

12 Break remaining groups into smaller groups or individual costs for use as factors.

Is linear model sufficient? — No / Yes

(approx. 5 min.)

13 Generate second-order design table in JMP. (approx. 10 sec.)

Determine cost drivers by proportion of sum of squares.

14 Use rows which have not been run as new design table. (approx. 20 sec.)

D-2

D.3. PROCESS DETAILS

[1] *SAILS base model.* A SAILS™ base model includes all the required input files with data which will run to successful solution when submitted to one of the SAILS™ solver engines.

[2] *Extract costs from SAILS input files into Excel "Work Space".* The Pascal routine in stripfaf.exe is designed to extract both fixed and variable costs on the model's distribution centers and place them in a text file named fafout.txt. This file is opened in Excel as a tab delimited file and moved into the Excel workbook containing the "Work Space" for this CIAT application. The Excel macro "faf2wkspace" copies all nonempty rows in the "fafout" sheet and inserts them properly into the "Work Space" sheet.

[3] *Set upper and lower bounds on costs to be investigated.* Columns are specified in the "Work Space" sheet for the initial cost value (extracted from "fafout") and upper and lower bounds. Bounds should only be added for costs which will be investigated in the CIAT application. The analyst may type in upper and lower bounds directly or use Excel formulae to specify a percentage increase and decrease of the initial cost value. If cost bounds are individually set, the value in the initial cost column must be revised so that it gives the midpoint of the range.

[4] *Group costs; use groups as factors.* As upper bounds are specified, the number of cost factors for the CIAT application is tallied in a cell on row two of the "Work Space". If this number of costs is greater than 31, groups must be specified in the given column. Groups should be designated by whole numbers, beginning with group 1. Costs not included in the CIAT application should be denoted by a zero in the

group column. The second row of the "Work Space" also contains a cell which holds

the number of groups specified (the maximum value contained in the group column).

[5] *Generate design table in JMP for factors.* Use the number of groups or number of

costs from the top of the "Work Space" to generate the appropriate design in JMP.

To get JMP to generate a coded experimental design table, go through "Table ->

Design Experiment". Then select two-level design from the pop-up menu. Once the

design is selected from the pop-up menu's listing, click "Generate Design". At this

time it is a good idea to save the design table to a permanent file.

[6] *Bring coded design table into "Work Space" & calculate true settings for design*

runs. Run the "Export2Dsign" macro to create a properly formatted text copy of the

"Work Space" sheet with the name "DsignBkgrnd.txt". Bring a text copy of

"DsignBkgrnd.txt" into JMP along with its labels. Subset the design table so that all

numeric columns are selected and transpose this subset table so that the factor labels,

e.g. "x1", are at the beginning of each row. Change these labels so that x1 becomes 1,

x2 becomes 2 and so on. Then merge the subset table onto the text copy of the

"DsignBkgrnd.txt" by matching the group column with the labels on the design table

subset. This requires use of the "Table -> Join" operation. Save the

"DsignBkgrnd.txt" text file which now includes the coded factor settings for each run

from the design table—make sure to specify the text file option with column labels.

Open this new text file in Excel and move it into the workbook used for this CIAT

application. This can be done by running the "ImportFromDsign" macro. To

calculate the real cost settings for the first design run, use an Excel formula which

places the lower bound value in the appropriate cell of the "Real1" column if the

corresponding value in column "Row1" is less than zero, the upper bound value if greater than zero, and the initial value otherwise. Use fixed cell references in the formula where you reference the upper bound column, the lower bound column, and the initial value column (this is done in current versions of Excel by placing a "$" before the column reference). Then the fill commands can be used to place this formula in columns for each run and rows for each factor.

[7] *Create SAILS files for runs in design table.* Run the "Outputdoe" macro to create a file "doe.txt" which contains the real cost setting revisions for each run in the design table and is placed in the "SAILS/modelbldr/models" subdirectory. Run the Pascal routines in "chgfaf.exe" and "filecopy.exe", specifying the root file name for the SAILS™ base model and the number of copies (number of runs in the design table).

[8] *Load & launch SAILS runs.* Select the _.ecf files for each run in the design table and load them on the run stack in the SAILS™ run launcher interface. Then launch the entire collection.

[9] *Extract response values from SAILS _.min files & add to coded table in JMP.* Run the Pascal routine in "stripmin.exe" if you are only interested in collecting the optimal system cost value. Specify the root file name and number of copies as before. This program extracts the total system cost from each of the specified runs and places them in the file "minout.txt" labeled and ordered by run number. If you also want to collect a list of DCs with positive throughput in the solution, run "stripm2.exe" instead. Open the "minout.txt" file in JMP as a tab delimited text file with no labels or headings. Merge this file with the design table, using the "Table -> Join" operation. If the row

order has been retained as the run order, the tables can be merged by row number. Otherwise, use the option to match the columns containing the run numbers.

[10]*Analyze model w/linear least squares regression.* Analyze the model in JMP. Select the new cost column as the Y variable and the factor columns as the X variables in the model. If the design table is second-order, use the "Effects" drop down menu to specify a response surface model. Determine which factors are drivers according to their relative contributions to the model sum of squares. Evaluate the presence of curvature by looking at the cost error of prediction at the center point or by including an appropriate number of center point rows in the model and considering the R^2 value.

[11]*Drop groups which have relatively small sum of squares.* Although determining which cost groups to drop is a subjective judgment, it will not be unusual to see eighty to ninety percent of the sum of squares contributed by a small collection of the groups.

[12]*Break remaining groups into smaller groups or individual costs for use as factors.* It is possible to break into individual costs with as many as thirty-one cost factors, with a thirty-two run Plackett-Burman experimental design. However, it may require fewer total runs if another grouped stage is performed before breaking into individual costs. This decision is also a matter for subjective analyst choice.

[13]*Generate second-order design table in JMP.* This component is similar to the steps in block [5]. The first difference is that the "Response Surface" option is selected from the pop-up menu rather than the "Two-Level Design" option used in [5]. Axial points should be specified as "face-centered" so that the only cost settings required are the upper and lower bounds and midpoint.

[14]*Use rows which have not been run as new design table.* This requires a subsetting

table operation. If the tables are small enough, the rows may be manually selected

before subsetting the table. If the tables are very large, it may be preferable to join the

tables by the pattern column first, keeping non-matched rows from both tables. Then

create a column in this combined table with a formula to contain a one for rows with

the same values in both pattern columns and a zero otherwise. Then the appropriate

rows can be selected for the subset table by using the "Rows -> Select -> Where..."

operation.

Finally, if desired, the response surface can be displayed in a three-dimensional

spinning plot platform in JMP. In order to generate the appropriate x-y grid points, use

the "Grdtmxy.jmp" file provided in the "Templats" folder. Enter -1 and +1 in the spaces

provided for the start grid and stop grid values, respectively. Next, add rows to the table

in a proportion which matches the density of spacing you want in your graph—so if you

want fifty points sampled over the 2×2 grid specified, add fifty rows. Return to the output

page which contains the response surface analysis results (obtained as a result of your "Fit

Model" operation). Save the prediction formula to the underlying data table using the

option given in the "$" menu in the lower left corner of the output page. View this

prediction formula by selecting the column heading, using the "Cols -> Column Info"

operation, and double clicking in the lower left corner where a small preview of the

formula is displayed. To select the entire formula, click on all portions of the formula

while holding the shift key. Copy the formula using the "Edit" menu option. Now return

to the grid table and paste this formula into the column given for the response values,

using the same steps required to copy the formula. Finally, select the "Spinning Plot"

option in the "Graph" menu and specify the x, y, and response columns to be plotted. The buttons provided to the left of the plot allow you to change your viewing perspective in many ways. Once you obtain a view which you would like to incorporate into a presentation in some other media, use the "Edit -> Journal" option to send the window contents to a journal page which can be saved in a rich text format.

APPENDIX E: EXCEL USER INTERFACE AND MACROS

E.1. USER INTERFACE

The CIAT user interface is a specially structured Excel workbook. The workbook contains a "Work Space" sheet which is the basis of each stage in a CIAT application. The initial format of the "Work Space" sheet is shown in Figure E.1.

Number of factors	0		Number of Groups	0		
File	SAILS ID	Cost Type	Initial Cost	Upper Bound	Lower Bound	Group
BOTTOM OF TABLE						

Fig. E.1. Excel "Work Space" format at start of CIAT application.

Once the costs are extracted from the base SAILS™ model, the text file containing this data is added as another sheet in the workbook, and the rows containing the cost data are copied and inserted into the table by one of the following Excel macros. Next, the analyst inserts the appropriate upper and lower bounds and adjusts the initial cost to their midpoint if necessary. The number of costs with bounds specified is shown as the "Number of factors". If grouping is required, the analyst numbers the costs according to their group membership. Any costs not included are designated with a zero in the group column.

Once the number of cost factors or group factors is used to generate the appropriate experimental design table in JMP, the coded factor settings are merged to a text copy of this table so that the -1, 0, and +1 settings appear on the appropriate rows (i.e. matched according to group or individual cost factor). This extended table is also opened and added to the workbook. In this sheet, the true cost settings are calculated for each cost factor and all runs in the design, by using the lower bound when a -1 appears in the coded column, the initial cost when a 0 appears, and the upper bound when a +1

appears. Finally, the text file "doe.txt" is created from this worksheet—for use by "chgfaf.exe" to create the modified _.faf files for input to SAILS™. Copies of the "Work Space" sheet and the sheet used as "doe.txt" can be saved in the workbook with a new name so that the analyst has a record of the stages completed in this CIAT application. The following sections contain copies of each of the macros included in the workbook to automate several of the required spreadsheet operations.

E.2. EXCEL MACROS

E.2.1.1. Role of the "Importfaf" Macro.

This macro opens the text file "FAFOUT.TXT", generated by the "stripfaf.exe" program, and moves it into our workbook as an additional sheet. Note that the workbook name must be inserted in place of "PFSWORK.xls" if the analyst wishes to use this macro.

E.2.1.2. "Importfaf" Code.

```
'
' Importfaf Macro
' Macro recorded 1/6/97 by Angela Giddings
'
'
Sub importfaf()
    ChDir "C:\SAILS\modlbldr\models"
    Workbooks.OpenText Filename:="C:\SAILS\modlbldr\models\FAFOUT.TXT", _
        Origin:=xlWindows, StartRow:=1, DataType:=xlFixedWidth, _
        FieldInfo:=Array(Array(0, 1), Array(3, 1), Array(15, 1), Array(23, 1))
    Sheets("FAFOUT").Move Before:=Workbooks("PFSWORK.xls").Sheets(1)
End Sub
```

E.2.2.1. Role of the "faf2Wkspace" Macro.

This macro inserts the cost data rows from the "FAFOUT" sheet to the "Work Space" sheet, immediately above the "BOTTOM OF TABLE" cell.

E.2.2.2. "faf2Wkspacef" Code.

```
'
'
' faf2Wkspace Macro
' Macro recorded 1/6/97 by Angela Giddings
'
'
Sub faf2wkspace()

    Sheets("Work Space").Select
    Range("A4").Select

    Sheets("FAFOUT").Select
    Range("A2").Select
    While ActiveCell <> ""
        ActiveCell.Rows("1:1").EntireRow.Select
        Selection.Copy
        Sheets("Work Space").Select
        ActiveWindow.ScrollRow = 1
        ActiveCell.Rows("1:1").EntireRow.Select
        Selection.Insert Shift:=xlDown
        ActiveCell.Offset(1, 0).Range("A1").Select
        Sheets("FAFOUT").Select
        ActiveCell.Offset(1, 0).Range("A1").Select
    Wend
    Application.CutCopyMode = False

End Sub
```

E.2.3.1. Role of the "Export2Design" Macro.

This macro names the current sheet (which should be the "Work Space" sheet or a copy of it) "DesignBkgrnd", removes the top two rows and the first row beneath the headings (which will be a header row from the FAFOUT sheet), and removes all rows with a "0" value in the group column.

E.2.3.2. "Export2Design" Code.

```
'
' Export2Design Macro
' Macro recorded 1/6/97 by Angela Giddings
'
'
```

```
Sub Export2Design()

    Sheets.Add
    ActiveSheet.Name = "DesignBkgrnd"

    Sheets("Work Space").Select
    Application.Goto Reference:="WorkArea"
    Selection.Copy
    Sheets("DesignBkgrnd").Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False


    Range("A1").Select
    Cells.Select
    Application.CutCopyMode = False
    Selection.Copy
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False

    While ActiveCell <> ""
        ActiveCell.Offset(0, 6).Range("A1").Select
        If ActiveCell = "0" Then
            ActiveCell.Rows("1:1").EntireRow.Select
            Selection.Delete Shift:=xlUp
        Else ActiveCell.Offset(1, -6).Range("A1").Select
        End If
    Wend

    ActiveWorkbook.Save
    ActiveWorkbook.SaveAs Filename:="C:\JMP\DsignBkgrd.txt", _
        FileFormat:=xlText, CreateBackup:=False

End Sub
```

E.2.4.1. *Role of the "ImportFromDesign" Macro.*

This macro opens the file "DsignBkgrd.txt" from the JMP folder where it was

appended with the coded factor levels. Next, the macro moves the text file into the

"PFSWORK.xls" workbook as an additional sheet. If the current CIAT project workbook

has a different name, the analyst must replace "PFSWORK.xls" with this name in the code.

E.2.4.2. "ImportFromDesign" Code.

```
'

' ImportFromDsign Macro

' Macro recorded 1/28/97 by Angela Giddings

'

'

Sub ImportFromDsign()

    ChDir "C:\JMP"

    Workbooks.OpenText Filename:="C:\JMP\DsignBkgrd.TXT", Origin:= _

        xlWindows, StartRow:=1, DataType:=xlDelimited, TextQualifier _

        :=xlDoubleQuote, ConsecutiveDelimiter:=False, Tab:=True, _

        Semicolon:=False, Comma:=False, Space:=False, Other:=False, _

        FieldInfo:=Array(Array(1, 1), Array(2, 1), Array(3, 1), Array(4, 1), Array _

        (5, 1), Array(6, 1), Array(7, 1), Array(8, 1), Array(9, 1), Array(10, 1), Array( _

        11, 1), Array(12, 1), Array(13, 1), Array(14, 1), Array(15, 1), Array(16, 1), _

        Array(17, 1))

    Sheets("DsignBkgrd").Move Before:=Workbooks("PFSWORK.xls").Sheets(1)

End Sub
```

E.2.5.1. Role of the "Outputdoe" Macro.

This macro works with the sheet that contains the coded factor settings for each cost in columns labeled "Row1", "Row2", and so on. This worksheet should also have the calculated true cost settings in columns labeled "Real1", "Real2", etc. The macro copies all cells in the sheet and pastes in their values so that no information is lost as some cells are deleted. Next the macro deletes all columns in the sheet except for the columns labeled "File", "SAILS ID", and "Cost Type" along with any "Real" cost setting columns. Finally, this sheet is saved as the text file "doe.txt" in the subdirectory "C:\SAILS\modelbldr\models" where the "chgfaf.exe" program uses it to create the modified _.faf files for each run in the design.

E.2.5.2. "Outputdoe" Code.

```
'
' Outputdoe Macro
' Macro recorded 1/6/97 by Angela Giddings
'
'
Sub Outputdoe()
    Range("A1").Select
    Cells.Select

    Application.CutCopyMode = False
    Selection.Copy
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, _
        SkipBlanks:=False, Transpose:=False

    While ActiveCell <> "Real1"

        If ActiveCell <> "Real1" Then
        If ActiveCell <> "File" Then
        If ActiveCell <> "SAILS ID" Then
        If ActiveCell <> "Cost Type" Then
            ActiveCell.Columns("A:A").EntireColumn.Select
            Selection.Delete Shift:=xlToLeft
```

```
        ActiveCell.Select
        ActiveCell.Offset(0, -1).Range("A1").Select

    End If
    End If
    End If
    End If
    ActiveCell.Offset(0, 1).Range("A1").Select
Wend
ActiveSheet.Name = "doe"
ActiveWorkbook.Save
Rows("1:1").Select
Selection.Delete Shift:=xlUp

ActiveWorkbook.SaveAs Filename:="C:\SAILS\modlbldr\models\doe.txt", _
    FileFormat:=xlText, CreateBackup:=False

End Sub
```

APPENDIX F: PASCAL ROUTINES

F.1. STRIPFAF

F.1.1. Purpose and Implementation.

The stripfaf.exe program is the executable file created once the following Turbo Pascal program stripfaf.pas is compiled. This program reads the _.faf file with the input root file name found in its resident subdirectory, and formats relevant fields (i.e., the SAILS™ id number for each distribution center, a label to denote the cost type for that line, and the fixed or variable cost value associated with that SAILS™ id number) for output to a text file which will be used in the Excel "Work Space" (the CIAT user interface).

The executable file should be invoked from the DOS prompt in its resident subdirectory with the command "stripfaf <root file name> <number of file copies>". The second parameter is actually not necessary, but it is included for consistency with the other Turbo Pascal programs.

F.1.2. Stripfaf.pas Code.

```
Program FAF;

uses CRT;

const
    twosp = ' ';
    fivesp = '     ';
    vLabel = ' variable ';
    fLabel = ' fixedcst ';

var
  FAFfile : text;
  FAFout : text;
  S : string[80];
  header : string[10];
```

F-1

```
    fixed : string[10];
    variable : string[10];
    SAILSID : string[10];
    fileID : string[5];
    oldName : string[10];


begin {program}

 ClrScr;
 if ParamCount <> 2 then
    begin
      Writeln('Input root file name and number of copies.');
      Halt;
    end;

   oldName:=Concat(ParamStr(1), '.faf');
   Assign(FAFfile, oldName);
   Assign(FAFout, 'FAFout.txt');
   Reset(FAFfile);
   Rewrite(FAFout);

   While not Eof(FAFfile) do
     begin {while not eof FAFfile 1}
       Readln(FAFfile, S);
      if (S = 'WHSECPTY') then
        begin {if S = 'WHSECPTY'}
           While not Eof(FAFfile) do
             begin {while not eof FAFfile 2}
                header:=Copy(S,1,10);
                fixed:=Copy(S,21,10);
                variable:=Copy(S,61,10);
                SAILSID:=Copy(S,77,4);
                FileID:='FAF';
                if not(header='          ') then
                    Writeln(FAFout, fileID,twosp,header);
                if header='          ' then
                 begin
                    Writeln(FAFout, fileID,twosp,SAILSID,fivesp,fLabel,fixed);
                    Writeln(FAFout, fileID,twosp,SAILSID,fivesp,vLabel,variable);
                 end;
                Readln(FAFfile,S);
            end; {while not eof FAFfile 2}

             header:=Copy(S,1,10);
```

```
            fixed:=Copy(S,21,10);
            variable:=Copy(S,61,10);
            SAILSID:=Copy(S,77,4);
            FileID:='FAF';
            if not(header='        ') then
               Writeln(FAFout, fileID,twosp,header);
            if header='        ' then
               begin
                  Writeln(FAFout, fileID,twosp,SAILSID,fivesp,fLabel,fixed);
                  Writeln(FAFout, fileID,twosp,SAILSID,fivesp,vLabel,variable);
               end;

      end; {if S = 'WHSECPTY'}
   end; {while not eof FAFfile 1}

 Close(FAFfile);
 Close(FAFout);

 Writeln;
 Write('Press ENTER...');
 Readln;

end.
```

F.2. CHGFAF

F.2.1. Purpose and Implementation.

The chgfaf.exe program is the executable file created once the following

Turbo Pascal program chgfaf.pas is compiled. This program reads the _.faf file with the

input root file name and the doe.txt file found in its resident subdirectory. The doe.txt file

is a text file which contains the new settings for the fixed and variable costs in all runs of

the chosen experimental design. This file is created and formatted in Excel after the coded

design table is obtained from JMP.

This program creates a _.faf file for each run specified in doe.txt and names them <root file name>1.faf, <root file name>2.faf, …, <root file name><number of file copies>.faf. These files contain the cost settings as specified by the design.

The executable file should be invoked from the DOS prompt in its resident subdirectory with the command "chgfaf <root file name> <number of file copies>".

F.2.2. Chgfaf.pas Code.

```
Program RunChangeFAF;

uses CRT;

 var
   RootName : string[6];
   max, where : integer;
   logfile : text;

Procedure ChangeFAF(name : string; max : integer);

 type
   fixedPtr = ^fixedRec;
   fixedRec = record
     fileId : string[3];
     SAILSid : string[4];
     costType : string[8];
     costStr : string[10];
     Prev, Next : fixedPtr;
   end;
   variaPtr = ^variaRec;
   variaRec = record
     fileId : string[3];
     SAILSid : string[4];
     costType : string[8];
     costStr : string[5];
     Prev, Next : variaPtr;
   end;

   linePtr = ^lineRec;
   lineRec = record
     Strng : string[100];
     Prev, Next : linePtr;
```

```
      end;


   var
     fixed : fixedRec;
     variable : variaRec;
     firstFxd, prevFxd, currentFxd, lastFxd : fixedPtr;
     firstVar, prevVar, currentVar, lastVar : variaPtr;
     fixedCst : real;
     variaCst : real;
     fixedS : string[10];
     variaS : string[5];
     oldFile, newFile, RunFile : text;
     i, j : integer;
     inputFName, newFName : string[12];
     fId : string[3];
     SId : string[5];
     cstType : string[9];
     line : lineRec;
     firstLn, prevLn, currentLn, lastLn : linePtr;
     Jchr : string[2];
     SAILSchk : string[6];
     seekingF, seekingV : boolean;


{*********************}
Procedure InitiateRunFiles(name : string; max : integer);
   var
     S : string[100];
     redflag : boolean;
     j : integer;

begin {procedure InitiateRunFiles}

     inputFName:=Concat(name, '.faf');
     Assign(oldFile, inputFName);
     Reset(oldFile);

   for j := 1 to max do
     begin
       Str(j, Jchr);
       newFName:= Concat(name, Jchr, '.faf');
       Assign(newFile, newFName);
       Rewrite(newFile);
        {Copy unchanged portion of FAF file}
```

```
redflag:=false;
While not redflag do
  begin
    Readln(oldFile, S);
    if S = 'WHSECPTY' then
      begin
       redflag:=true;
       Writeln(newFile,S);
       Close(newFile);
       if j < max then
          Reset(oldFile);
      end;
    if S <> 'WHSECPTY' then
      begin
       Writeln(newFile, S);
      end;
  end; {while not redflag}
end; {for j}

{read last portion of oldFile into linked list for modification}
firstLn := nil;
   While not Eof(oldFile) do
    begin
      if firstLn = nil then
        begin
          New(currentLn);
          with currentLn^ do
            Readln(oldFile, Strng);
          currentLn^.Next := nil;
          currentLn^.Prev := nil;
          firstLn := currentLn;
          lastLn := currentLn;
        end
      else
        begin
          prevLn := lastLn;
          New(currentLn);
          with currentLn^ do
            Readln(oldFile, Strng);
          prevLn^.Next := currentLn;
          currentLn^.Next := nil;
          currentLn^.Prev := prevLn;
          lastLn := currentLn;
        end;
    end; {while not eof}
```

```pascal
      Close(oldFile);

      {print contents of list}
      Writeln(logFile, 'Contents of old list:');
      currentLn:=firstLn;
      while currentLn<>nil do
       begin
        with currentLn^ do
          Writeln(logFile, Strng);
         currentLn:=currentLn^.next;
        end; {print loop}


end; {procedure InitiateRunFiles}


{*******************}
begin {ChangeFAF procedure}

  Assign(logFile, 'pasLog.txt');
  Rewrite(logFile);

  InitiateRunFiles(name, max);
  Writeln;

  {Prepare RunFile}
   Assign(RunFile, 'doe.txt');

 for j:=1 to max do
  begin {for j:=1 to max}

     firstFxd:=nil;
     firstVar:=nil;

   {read data from RunFile for run j
   into fixed & variable lists}
     Reset(RunFile);
     while not Eof(RunFile) do
       begin {while not eof}
         Read(RunFile, fId, SId, cstType);
         Delete(SId, 1, 1);
         Delete(cstType, 1, 1);

         if cstType = 'fixedcst' then
```

```
begin {if fixed cost}
  i:=1;
  while i < j do
    begin
      Read(RunFile, fixedCst);
      Writeln(logFile, SId, 'Reading past fixed cost ', fixedCst);
      fixedCst:=0;
      i:=i+1;
    end;
  Readln(RunFile, fixedCst);
  Writeln(logFile, SId, 'Set on fixed cost ', fixedCst);
  Str(Round(fixedCst), fixedS);

  while Length(fixedS) < 10 do
    begin {while}
      insert(' ', fixedS, 1);
    end;


  if firstFxd=nil then
    begin {if fixed list empty}
      New(currentFxd);
      with currentFxd^ do
        begin
          costStr:=fixedS;
          fileId:=fId;
          SAILSid:=SId;
          costType:=cstType;
        end;
      currentFxd^.Next := nil;
      currentFxd^.Prev := nil;
      firstFxd := currentFxd;
      lastFxd := currentFxd;
    end {if fixed list empty}
  else
    begin {if fixed list not empty}
      prevFxd := lastFxd;
      New(currentFxd);
      with currentFxd^ do
        begin
          costStr:=fixedS;
          fileId:=fId;
          SAILSid:=SId;
          costType:=cstType;
        end;
```

```
              prevFxd^.Next := currentFxd;
              currentFxd^.Next := nil;
              currentFxd^.Prev := prevFxd;
              lastFxd := currentFxd;
            end;  {if fixed list not empty}
       end;  {if fixed cost}


   if cstType = 'variable' then
     begin  {if variable}
       i:=1;
       while i < j do
        begin
          Read(RunFile, variaCst);
          Writeln(logFile, SId, 'Reading past variable cost ', variaCst);
          variaCst:=0;
          i:=i+1;
         end;
       Readln(RunFile, variaCst);
       Writeln(logFile, SId, 'Set on variable cost ', variaCst);
       Str(Round(variaCst), variaS);

       while Length(variaS) < 5 do
        begin
          Insert(' ',variaS, 1);
         end;

       if firstVar=nil then
         begin  {if variable list empty}
           New(currentVar);
           with currentVar^ do
            begin
              costStr:=variaS;
              fileId:=fId;
              SAILSid:=SId;
              costType:=cstType;
             end;
           currentVar^.Next := nil;
           currentVar^.Prev := nil;
           firstVar := currentVar;
           lastVar := currentVar;
         end {if variable list empty}
        else
          begin  {if variable list not empty}
            prevVar := lastVar;
```

```
            New(currentVar);
            with currentVar^ do
              begin
                costStr:=variaS;
                fileId:=fId;
                SAILSid:=SId;
                costType:=cstType;
              end;
            prevVar^.Next := currentVar;
            currentVar^.Next := nil;
            currentVar^.Prev := prevVar;
            lastVar := currentVar;
          end;  {if variable list not empty}
        end; {if variable cost}


    end;  {while not eof}


  {debugging}
   currentFxd:=firstFxd;
   while currentFxd <> nil do
    begin
     with currentFxd^ do
       Writeln(logFile,'j=',j,'F: ',{fileId,'*',} SAILSid,'*', costType,'*', costStr,'*',
Length(costStr));
       currentFxd:=currentFxd^.next;
     end;{fixed debugging while loop}
     currentVar:=firstVar;
     while currentVar <> nil do
      begin
       with currentVar^ do
         Writeln(logFile,'j=',j,'V: ',{fileId,'*',} SAILSid,'*', costType,'*', costStr,'*',
Length(costStr));
         currentVar:=currentVar^.next;
       end;{variable debugging while loop}


     {prepare newFile for run j for modified last part}
     Str(j, Jchr);
     newFName:= Concat(name, Jchr, '.faf');
     Assign(newFile, newFName);
     Append(newFile);

     currentLn:=firstLn;
     while currentLn <> nil do
       begin {while old list not at end}
```

F-10

```
with currentLn^ do
 begin {with current line of old list}
   SAILSchk:=Copy(Strng,77,4);
   currentFxd:=firstFxd;
   seekingF:=true;
   currentVar:=firstVar;
   seekingV:=true;

 {check fixed list}
   while (currentFxd <> nil) and (seekingF) do
    begin
     with currentFxd^ do
      begin
       if SAILSchk=SAILSid then
        begin
         Delete(Strng, 21, 10);
         Insert(costStr, Strng, 21);
         seekingF:=false;
        end;
      end; {with current fixed entry}
     currentFxd:=currentFxd^.next;
    end;{check of fixed list}

 {output message fixed cost not used}
  if seekingF then
    Writeln(logFile,'Fixed cost for SAILS id ', SAILSchk, ' not changed.');

 {check variable list}
   while (currentVar <> nil) and (seekingV) do
    begin
     with currentVar^ do
      begin
       if SAILSchk=SAILSid then
        begin
         Delete(Strng, 66, 5);
         Insert(costStr, Strng, 66);
         seekingV:=false;
        end;
      end; {with current variable entry}
     currentVar:=currentVar^.next;
    end;{check of variable list}

 {output message variable cost not used}
  if seekingV then
    Writeln(logFile,'Variable cost for SAILS id ', SAILSchk, ' not changed.');
```

```pascal
        Writeln(newFile, Strng);
        currentLn:=currentLn^.next;

      end;{with current line of old list}

   end; {while old list not at end}

{debugging}

 currentLn:=firstLn;
 while currentLn<>nil do
  begin
   with currentLn^ do
    begin
     Writeln(logFile,'j=',j,' old: ');
     Writeln(logFile, Strng);
    end;
    currentLn:=currentLn^.next;
   end; {old list debugging while loop}

   Close(newFile);
 end; {for j:=1 to max}


 Close(RunFile);


end; {ChangeFAF procedure}

begin {Program}

 ClrScr;
 if ParamCount <> 2 then
   begin
     Writeln('Input root file name and number of copies.');
     Halt;
   end;



 Writeln('In program');
 RootName:= ParamStr(1);
 Val(ParamStr(2), max, where);
```

```
     ChangeFAF(RootName, max);
     Write('End program.  Press ENTER...');
     Readln;
   end.
```

F.3. FILECOPY

F.3.1. Purpose and Implementation.

The filecopy.exe program is the executable file created once the following Turbo

Pascal program filecopy.pas is compiled. This program copies all input SAILS files except

for the _.faf file the specified number of times and names them <root file name>1.xxx,

<root file name>2.xxx, ..., <root file name><number of file copies>.xxx, where the xxx

represents the file extensions for SAILS input.

The executable file should be invoked from the DOS prompt in its resident

subdirectory with the command "filecopy <root file name> <number of file copies>".

F.3.2. Filecopy.pas Code.

```
Program CopyFile;

uses CRT;

type
   fileType = (cdf, ebf, ecf, mcf, run, tdf, sdf, sgf);
   extType = array [cdf..{, ebf, ecf, mcf, run, tdf, sdf,} sgf] of string[4];

var
   SourceFile,
   DestFile : file;
   RecordsRead : integer;
   Buffer : array [1..1000] of byte;
   newName, oldName : string [12];
   S : string [2];
   ctr,max,where : integer;
   extCtr : fileType;
   extension : extType;

begin
```

```
ClrScr;
if ParamCount <> 2 then
  begin
    Writeln('Input root file name then number of copies.');
    Halt;
  end;

extension[cdf]:='.cdf';
extension[ebf]:='.ebf';
extension[ecf]:='.ecf';
extension[mcf]:='.mcf';
extension[run]:='.run';
extension[tdf]:= '.tdf';
extension[sdf]:='.sdf';
extension[sgf]:='.sgf';

for extCtr:=cdf to sgf do
  begin
    oldName:=Concat(ParamStr(1), extension[extCtr]);
    Val(ParamStr(2), max, where);
    for ctr := 1 to max do
      begin {Make specified copies of this file type}
        Str(ctr,S);
        newName:=Concat(ParamStr(1), S, extension[extCtr]);
        Assign(SourceFile, oldName);
        {I-}
        Reset(SourceFile, 1);
        {I+}
        if IOresult <> 0 then
          begin
            Writeln('Input file not found.');
            Halt;
          end;

        Writeln(' . = 1,000 bytes copied.');

        Assign(DestFile, newName);
        Rewrite(DestFile, 1);
        BlockRead(SourceFile, Buffer, SizeOf(Buffer), RecordsRead);

        while RecordsRead > 0 do
          begin
            Write('.');
            BlockWrite(DestFile, Buffer, RecordsRead);
            BlockRead(SourceFile, Buffer, Sizeof(Buffer), RecordsRead);
```

```
        end;

      Close(SourceFile);
      Close(DestFile);
    end;


  Writeln;
  Write('Press ENTER...');
  Readln;
  end;

end.
```

F.4. STRIPMIN

F.4.1. Purpose and Implementation.

The stripmin.exe program is the executable file created once the following Turbo

Pascal program, stripmin.pas, is compiled. This program reads the _.min SAILS output

files, with the specified root file name, found in its resident subdirectory and formats

relevant fields (i.e. the run number and the optimal objective function value for that run)

for output to a text file which will be used to provide response values for the JMP

experimental design table generated for this stage in CIAT implementation.

The executable file should be invoked from the DOS prompt in its resident

subdirectory with the command "stripmin <root file name> <number of file copies>".

F.4.2. Stripmin.pas Code.

```
Program stripMIN;

uses CRT;

var
  MINfile : text;
  MINout : text;
  LogFile : text;
  S : string[180];
  header : string[30];
```

```pascal
  totalCst : string[20];
  SAILSID : string[10];
  fileID : string[5];
  oldName : string[12];
  j, max, where : integer;
  Jchr : string[2];



begin {program}
  ClrScr;
  if ParamCount <> 2 then
    begin
      Writeln('Input root file name and number of runs.');
      Halt;
    end;

  Assign(logFile, 'pasLog.txt');
  Rewrite(logFile);
  Assign(MINout, 'MINout.txt');
  Rewrite(MINout);
  Val(ParamStr(2), max, where);

  writeln(logFile, 'In program');


  for j:=1 to max do
    begin {for j:=1 to max}
      Str(j, Jchr);
      oldName:=Concat(ParamStr(1), Jchr, '.min');
      Assign(MINfile, oldName);
      Reset(MINfile);

      write(logFile, 'Reading file ',oldName);
      writeln(logFile, ', j=',j,' *',jchr,'*');

      header:=' ';
      repeat
        Readln(MINfile, S);
        header:=Copy(S, 10, 21);
      until header='SYSTEMWIDE TOTAL COST';

      Write(MINout, 'Run ', Jchr, ':');
      Writeln(MINout, copy(S,80,25));
      write(logFile, 'Run ', Jchr, ':');
```

```
        writeln(logFile, copy(S,80,25));

        Close(MINFile);

      end; {for j:=1 to max}
      Close(MINout);
      Close(logFile);

end. {program}
```

F.5. STRIPM2

F.5.1. Purpose and Implementation.

The stripm2.exe program is the executable file created once the following Turbo

Pascal program, stripm2.pas, is compiled. This program acts in exactly the same fashion

as the stripmin.exe program, except that it also collects the names of all DCs with a

positive throughput in the _.min report for each run.

The executable file should be invoked from the DOS prompt in its resident

subdirectory with the command "stripm2 <root file name> <number of file copies>".

F.5.2. Stripm2.pas Code.

```
Program stripMIN;

uses CRT;

var
   MINfile : text;
   MINout : text;
   LogFile : text;
   S : string[180];
   header : string[30];
   throughtxt : string[15];
   throughput : integer;
   totalCst : string[20];
   SAILSID : string[10];
   fileID : string[5];
   oldName : string[12];
   j, max, where : integer;
```

```pascal
     Jchr : string[2];



begin {program}
  ClrScr;
  if ParamCount <> 2 then
     begin
       Writeln('Input root file name and number of runs.');
       Halt;
     end;

    Assign(logFile, 'pasLog.txt');
    Rewrite(logFile);
    Assign(MINout, 'MINout.txt');
    Rewrite(MINout);
    Val(ParamStr(2), max, where);

    writeln(logFile, 'In program');


    for j:=1 to max do
      begin {for j:=1 to max}
        Str(j, Jchr);
        oldName:=Concat(ParamStr(1), Jchr, '.min');
        Assign(MINfile, oldName);
        Reset(MINfile);

        write(logFile, 'Reading file ',oldName);
        writeln(logFile, ', j=',j,' *',jchr,'*');

        header:=' ';
        repeat
           Readln(MINfile, S);
           header:=Copy(S, 10, 21);
        until header='SYSTEMWIDE TOTAL COST';

        Write(MINout, 'Run ', Jchr, ':');
        Write(MINout, copy(S,80,25));
        write(logFile, 'Run ', Jchr, ':');
        writeln(logFile, copy(S,80,25));
repeat
     Readln(MINfile, S);
     header:=Copy(S, 1, 17);
until header=' REPORT #: 56-05A';
```

```
writeln(logFile, header);
Readln(MINfile, S);
Readln(MINfile, S);
Readln(MINfile, S);
Readln(MINfile, S);

repeat
    Readln(MINfile, S);
    header:=Copy(S, 1, 20);
    throughtxt:=Copy(S, 25, 14);
    write(logFile, header);
    write(logFile, throughtxt);
    Val(throughtxt,throughput,where);
    writeln(logFile, '*',throughput);
    if throughput > 0 then write(MINout, header);
until header='0   TOTALS:        ';

writeln(MINout);

    Close(MINFile);

    end; {for j:=1 to max}
    Close(MINout);
    Close(logFile);

end. {program}
```

BIBLIOGRAPHY

Apostol, T.M. *Mathematical Analysis*. Reading, MA: Addison-Wesley, 1974.

Arntzen, B. C., G. G. Brown, T. P. Harrison, and L. Trafton. "Global Supply Chain Management at Digital Equipment Corporation," *Interfaces, 25:* 69-93 (January-February 1995).

Barlaz, A. Logistics Consultant, PFS Logistics Consulting, Dallas, TX. Personal interview. 28 June 1996.

Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows: Second Edition*. New York: John Wiley & Sons, Inc., 1990.

Booth, K. H. V. and D. R. Cox. "Some Systematic Supersaturated Designs," *Technometrics, 4:* 489-495 (1962).

Box, G. P. and N. R. Draper. *Empirical Model-Building and Response Surfaces*. New York: John Wiley & Sons, Inc., 1987.

Box, G. P., W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. New York: John Wiley & Sons, Inc., 1978.

Breitman, R. L. and J. M. Lucas. "PLANETS: A Modeling System for Business Planning," *Interfaces, 17:* 94-106 (January-February 1987).

Brown, G. G., G. W. Graves, and M. D. Honczarenko. "Design and Operation of a Multicommodity Production/Distribution System Using Primal Goal Decomposition," *Management Science, 33:* 1469-1480 (November 1987).

Clarkson, W. Vice President, Logistics Consulting, PFS Logistics Consulting, Dallas, TX. Personal interview. 26 June 1996.

Clarkson, W. Vice President, Logistics Consulting, PFS Logistics Consulting, Dallas, TX. Personal Correspondence. 12 February 1997.

Concepts: A Handbook for SAILS™ Users, Volume 1. Alexandria, VA: INSIGHT, Inc., 1995.

Deviprasad, J. Sr. Logistics Consultant, PFS Logistics Consulting, Dallas, TX. Personal interview. 24 Sept 1996.

Francis, R. L. and J. A. White. *Facility Layout and Location: An Analytical Approach*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1974.

Ganeshan, R. and T. P. Harrison. "An Introduction to Supply Chain Management," http://silmaril.smeal.psu.edu/misc/supply_chain_intro.html, 22 May 1995.

Geoffrion, A. M. and G. W. Graves. "Multicommodity Distribution System Design by Benders Decomposition," *Management Science, 20:* 822-844 (January 1974).

Geoffrion, A. M. and R. Nauss. "Parametric and Postoptimality Analysis in Integer Linear Programming," *Management Science, 23:* 453-466 (January 1977).

Geoffrion, A. M. and R. F. Powers. "20 Years of Strategic Distribution System Design: An Evolutionary Perspective," Working Paper No. 431, Western Management Science Institute, University of California, Los Angeles CA, 1-25, November 1993. Also in *Interfaces, 25:* 105-127 (September - October 1995).

Graney, R. E. *An Optimization Methodology for Multi-Criteria Comparison Using Response Surface Methods and Mathematical Programming*. MS thesis,

AFIT/GST/OS/84M-11. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1984 (AD-A141169).

Haile, R. F. *The Use of Mathematical Programming and Response Surface Methodology in Optimizing the Airlift Force Structure in a Far Eastern Theater of Operation.* MS thesis, AFIT/GOR/OS/86D-8. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986 (AD-A185269).

Hansen, R. Sr. Manager, PFS Logistics Consulting, Dallas, TX. Personal interview. 26 June 1996.

Ishihara, B. K. *Evaluating Experimental Designs for Fitting Response Surfaces of Deterministic Models.* MS thesis, AFIT/GOR/OS/85D-10. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1985 (AD-A167604).

Johnson, K. E., K. W. Bauer, Jr., J. T. Moore, and M. Grant. "Metamodelling Techniques in Multidimensional Optimality Analysis for Linear Programming," *Mathematical and Computer Modelling, 23:* 45-60 (1996).

Kleijnen, J. P. C. "Review of Random and Group-Screening Designs," *Communications in Statistics - Theory and Methods, 16:* 2885-2900 (1987a).

Kleijnen, J. P. C. "Simulation with Too Many Factors: Review of Random and Group-Screening Designs," *European Journal of Operational Research, 31:* 31-36 (1987b).

Kleijnen, J. P. C. "Screening Designs for Poly-Factor Experimentation," *Technometrics, 17:* 487-493 (November 1975).

Lasdon, L. S. *Optimization Theory for Large Systems.* New York: Macmillan Publishing Co., Inc., 1970.

Love, R. F., J. G. Morris, and G. O. Wesolowsky. *Facilities Location: Models and Methods.* New York: Elsevier Science Publishing Co., Inc., 1988.

Mauro, C. A. *Efficient Identification of Important Factors in Large Scale Simulations: Technical Report No. 123-2.* U.S. Navy Contract No. N00014-84-C-0573. State College, PA: Desmatics, Inc., October 1986 (AD-A173 497).

Mauro, C. A. *A Comparison of Random Balance and Two-Stage Group Screening Designs - Part I: Technical Report No. 113-12.* Office of Naval Research Contract No. N00014-79-C-0650, Task No. NR 042-467. State College, PA: Desmatics, Inc., December 1983 (AD-A136 487).

Mauro, C. A. and K. C. Burns. *A Comparison of Random Balance and Two-Stage Group Screening Designs - Part II: Technical Report No. 113-14.* Office of Naval Research Contract No. N00014-79-C-0650, Task No. NR 042-467. State College, PA: Desmatics, Inc., February 1984 (AD-A137 994).

Myers, R. H. and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* New York: John Wiley & Sons, Inc., 1995.

Montgomery, D. C. *Design and Analysis of Experiments.* New York: John Wiley & Sons, Inc., 1976.

Moore, J. T. Individual Lecture on Benders' Decomposition. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, 16 July 1996.

Nauss, R. M. *Parametric Integer Programming*. Columbia, Missouri: University of Missouri Press, 1979.

Patel, M. S. "Group-Screening with More Than Two Stages," *Technometrics, 4:* 209-217, (May 1962).

Powers, R. F. "Optimization Models for Logistics Decisions," *Journal of Business Logistics, 10:* 106-121, (1989).

Satterthwaite, F. E. "Random Balance Experimentation," *Technometrics, 1:* 111-137 (May 1959).

Sengupta, S. and Turnbull, J. "Seamless Optimization of the Entire Supply Chain," *IIE Solutions, 28:* 28-33 (October 1996).

Smith, D. E. and C. A. Mauro. "Factor Screening in Computer Simulation," *Simulation, 38:* 49-54 (February 1982).

Smith, P. W. *A Methodology for Developing and Analyzing the Optimal Response of an Economic Criterion to Simultaneously Induced Multiple Events Using Input-Output, Mathematical Programming, and Response Surface Methodologies*. PhD dissertation. University of Alabama, Tuscaloosa AL, 1975 (AAG7613941).

Sparrow, K. J. *An Interactive Computer Package for Use with Simulation Models Which Performs Multidimensional Sensitivity Analysis by Employing the Techniques of Response Surface Methodology*. MS thesis, AFIT/GOR/OS/84D-12. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984 (AD-A151956).

Villareal, A. Sr. Logistics Consultant, PFS Logistics Consulting, Dallas, TX. Personal interview. 27 June 1996.

Ward, J. E. and R. E. Wendell. "Approaches to Sensitivity Analysis in Linear Programming," *Annals of Operations Research, 27:* 3-38 (1990).

Watson, G. S. "A Study of the Group Screening Method," *Technometrics, 3:* 371-388 (August 1961).

Winston, W. L. *Operations Research: Applications and Algorithms.* Belmont, California: Wadsworth Publishing Company, 1994.

VITA

1Lt Angela Pippin Giddings was born on 19 December 1970 in Kirksville, Missouri, to Loren and Patsy Pippin. She graduated from LaPlata High School in 1989 and began undergraduate studies at Rhodes College in Memphis, Tennessee. Through a cross-town ROTC program with Memphis State University, she pursued an Air Force commission. A summer internship at Oak Ridge National Laboratory in Oak Ridge, Tennessee, resulted in her first publication—"United States Emissions of Carbon Dioxide to the Earth's Atmosphere by Economic Activity", co-authored with Gregg Marland, in *Energy Systems and Policy*. With undergraduate requirements met and ROTC training completed in December 1993, she was commissioned on 18 December 1993. She graduated Cum Laude and Phi Beta Kappa with a Bachelor of Science degree in Mathematics in May 1994.

Her first assignment was to Air Force Recruiting Service, Randolph AFB, as a scientific analyst. In August 1995, she entered the Graduate Operations Research program in the School of Engineering at the Air Force Institute of Technology. Through her work in the first year of courses, she was inducted to Tau Beta Pi. Upon completion of the AFIT masters degree, she begins an assignment at the Office of Aerospace Studies, Kirtland AFB. She married Scott Giddings 10 May 1992, and they have a son, Aaron.

Permanent Address:

> Rt 1 Box 3

> LaPlata, MO 63549

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE February 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
A COST IMPACT ASSESSMENT TOOL
FOR PFS LOGISTICS CONSULTING

5. FUNDING NUMBERS

6. AUTHOR(S)

Angela P. Giddings, First Lieutenant, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology/ENS
2750 P Street
Wright-Patterson AFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/97M-08

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

PFS Logistics Consulting, A Division of PepsiCo, Inc.
14841 Dallas Parkway
Dallas, TX 75240

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution is Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Response surface methodology (RSM) is used for optimality analysis of the cost parameters in mixed integer linear programming. The mixed integer linear programming problems of interest are the large-scale problems in supply chain optimization—also known as facility location and allocation problems. Furthermore, this optimality analysis technique applies to optimality analysis of costs or right-hand-side elements in continuous linear programs and optimality analysis of costs in mixed or pure integer linear programs. A system which automates this process for supply chain optimization at PFS Logistics Consulting is also presented, along with description of its application and impact in their daily operations.

14. SUBJECT TERMS
Optimality Analysis, Response Surface Methodology, Design of Experiments, Group Screening, Mixed Integer Linear Programming, Supply Chain Optimization, Metamodels

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements*.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	-	Contract	PR	-	Project
G	-	Grant	TA	-	Task
PE	-	Program Element	WU	-	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	-	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	-	See authorities.
NASA	-	See Handbook NHB 2200.2.
NTIS	-	Leave blank.

Block 12b. Distribution Code.

DOD	-	Leave blank.
DOE	-	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	-	Leave blank.
NTIS	-	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.